

Exam Questions 1Z0-804

Java SE 7 Programmer II Exam

<https://www.2passeasy.com/dumps/1Z0-804/>



NEW QUESTION 1

Given:

```
Deque <String> myDeque = new ArrayDeque<String>();

myDeque.push("one");
myDeque.push("two");
myDeque.push("three");

System.out.println(myDeque.pop());
```

What is the result?

- A. Three
- B. One
- C. Compilation fails.
- D. The program runs, but prints no output.

Answer: A**Explanation:** push
void push(E e)Pushes an element onto the stack represented by this deque (in other words, at the head of this deque) if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an `IllegalStateException` if no space is currently available.This method is equivalent to `addFirst(E)`. pop

E pop()

Pops an element from the stack represented by this deque. In other words, removes and returns the first element of this deque.

This method is equivalent to `removeFirst()`. Returns:

the element at the front of this deque (which is the top of the stack represented by this deque)

Throws:

`NoSuchElementException` - if this deque is empty**NEW QUESTION 2**

Given the code fragment:

```
try {
    String query = "SELECT * FROM Employee WHERE ID=110";
    Statement stmt = conn.createStatement() ;
    ResultSet rs = stmt.executeQuery(query) ;           // Line 13
    System.out.println("Employee ID: " + rs.getInt("ID")); // Line 14
} catch (Exception se) {
    System.out.println("Error");
}
```

Assume that the SQL query matches one record. What is the result of compiling and executing this code?

- A. The code prints Error.
- B. The code prints the employee ID.
- C. Compilation fails due to an error at line 13.
- D. Compilation fails due to an error at line 14.

Answer: A**Explanation:** The code compiles fine.A: prints Error: `rs.next()` fehlt !! Fehlermeldung: Before start of result set mit `rs.next()` Aufruf : The code would run fine.`public int getInt(String columnName)` throws `SQLException`Retrieves the value of the designated column in the current row of this `ResultSet` object as an `int` in the Java programming language**NEW QUESTION 3**

Given:

```
public class DoubleThread {  
  
    public static void main(String[] args) {  
        Thread t1 = new Thread() {  
            public void run() {  
                System.out.print("Greeting");  
            }  
        }  
  
        Thread t2 = new Thread(t1); // Line 9  
  
        t2.run();  
  
    }  
}
```

Which two are true?

- A. A runtime exception is thrown on line 9.
- B. No output is produced.
- C. Greeting is printed once.
- D. Greeting is printed twice.
- E. No new threads of execution are started within the main method.
- F. One new thread of execution is started within the main method.
- G. Two new threads of execution are started within the main method.

Answer: CE

Explanation: Thread t2 is executed. Execution of T2 starts executionen of t1. Greeting is printed during theexecution of t1.

NEW QUESTION 4

What are two differences between Callable and Runnable?

- A. A Callable can return a value when executing, but a Runnable cannot.
- B. A Callable can be executed by a ExecutorService, but a Runnable cannot.
- C. A Callable can be passed to a Thread, but a Runnable cannot.
- D. A Callable can throw an Exception when executing, but a Runnable cannot.

Answer: AD

Explanation: The Callable interface is similar to Runnable, in that both are designed for classes whose instances arepotentially executed by another thread. A Runnable, however, does not return a result and cannot throw achecked exception.

NEW QUESTION 5

Which represents part of a DAO design pattern?

- A. interface EmployeeDAO { int getID();Employee findByID (intid); void update();void delete();}
- B. class EmployeeDAO { int getID() { return 0;}Employee findByID (int id) { return null;} void update () {}void delete () {}}
- C. class EmployeeDAO { void create (Employee e) {} void update (Employee e) {} void delete (int id) {}Employee findByID (int id) {return id}}
- D. interface EmployeeDAO { void create (Employee e); void update (Employee e); void delete (int id); Employee findByID (int id);}
- E. interface EmployeeDAO {void create (Connection c, Employee e); void update (Connection c, Employee e); void delete (Connection c, int id); Employee findByID (Connection c, int id);}

Answer: D

NEW QUESTION 6

Given the code fragment:

```
static public void otherMethod() {
    printFile("");
}

static public void printFile(String file) {
    try ( FileInputStream fis = new FileInputStream(file) ) {
        System.out.println (fis.read());
    } catch (IOException e) {
        printStackTrace();
    }
}
```

Why is there no output when otherMethod is called?

- A. An exception other than IOException is thrown.
- B. Standard error is not mapped to the console.
- C. There is a compilation error.
- D. The exception is suppressed.

Answer: C

Explanation: C: wenn printStackTrace() ohne Referenz auf das Exception object aufgerufen A : java.io.FileNotFoundException: wenn e.printStackTrace();
 The code compiles fine The line
 FileInputStream fis = new FileInputStream(file)) will fail at runtime since file is an empty string. Note:
 public void printStackTrace()
 Prints this throwable and its backtrace to the standard error stream.

NEW QUESTION 7

Which three are true?

- A. A setAutoCommit (False) method invocation starts a transaction context.
- B. An instance of Savepoint represents a point in the current transaction context.
- C. A rollback () method invocation rolls a transaction back to the last savepoint.
- D. A rollback () method invocation releases any database locks currently held by this connection object.
- E. After calling rollback (mysavepoint), you must close the savepoint object by calling mySavepoint.close() .

Answer: ABC

Explanation: A:The way to allow two or more statements to be grouped into a transaction is to disable the auto-commitmode. After the auto-commit mode is disabled, no SQL statements are committed until you call the methodcommit explicitly. All statements executed after the previous call to the method commit are included in thecurrent transaction and committed together as a unit.

Note:When a connection is created, it is in auto-commit mode. This means that each individual SQL statementis treated as a transaction and is automatically committed right after it is executed. (To be more precise, thedefault is for a SQL statement to be committed when it is completed, not when it is executed. A statement iscompleted when all of its result sets and update counts have been retrieved. In almost all cases, however, astatement is completed, and therefore committed, right after it is executed.)

B:The method Connection.setSavepoint, sets a Savepoint object within the current transaction. The
 Connection.rollback method is overloaded to take a Savepoint argument. When a transaction is rolled back toa savepoint all changes made after that savepoint are undone. C: calling the method rollback terminates a transaction and returns any values that were modified to theirprevious values. If you are trying to execute one or more statements in a transaction and get a SQLException, call the method rollback to end the transaction and start the transaction all over again.

NEW QUESTION 8

Given:

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

class Test {
    private static String REGEX = "\\Sto\\S|\\bo\\b";
    private static String INPUT = "Nice to see you,to,be fine.";
    private static String REPLACE = ",";

    public static void main(String[] args) {
        Pattern p = Pattern.compile(REGEX);
        Matcher m = p.matcher(INPUT);
        INPUT = m.replaceAll(REPLACE);
        System.out.println(INPUT);
    }
}
```

What is the result?

- A. Nice to see you,be fine
- B. Nice,see you,be fine
- C. Nice,see you, to, be fine
- D. Nice, see you, be fine
- E. Nice to see y, u, be fine

Answer: A

Explanation: The text ",to," is replaced by the ","

NEW QUESTION 9

Given that myfile.txt contains:

First
Second
Third

Given the following code fragment:

```
public class ReadFile02 {  
  
    public static void main(String[] args) {  
        String fileName1 = "myfile.txt";  
        String fileName2 = "newfile.txt";  
  
        try (    BufferedReader buffIn = new BufferedReader(new FileReader(fileName1));  
                BufferedWriter buffOut = new BufferedWriter(new FileWriter(fileName2)) ) {  
  
            String line = ""; int count = 1;  
            line = buffIn.readLine();  
            while (line != null) {  
                buffOut.write(count + ": " + line);  
                buffOut.newLine();  
                count++;  
                line = buffIn.readLine();  
            }  
        } catch (IOException e) {  
            System.out.println("Exception: " + e.getMessage());  
        }  
    }  
}
```

What is the result?

- A. new file.txt contains: 1: First2: Second3: Third
- B. new file.txt contains:1: First 2: Second 3: Third
- C. newfile.txt is empty
- D. an exception is thrown at runtime
- E. compilation fails

Answer: A

Explanation: For each line in the file myfile.text the line number and the line is written into newfile.txt.

NEW QUESTION 10

Which two codes correctly represent a standard language locale code?

- A. ES
- B. FR
- C. U8
- D. Es
- E. fr
- F. u8

Answer: AB

Explanation: Language codes are defined by ISO 639, an international standard that assigns two- and three-letter codes to most languages of the world. Locale uses the two-letter codes to identify the target language.

NEW QUESTION 10

Given:

```
import java.util.*;
public class SearchText {
public static void main(String[] args) {
Object[] array1 = new Object[3];
array1[0] = "foo";
array1[0] = 1;
array1[0] = 'a';
int index = Arrays.binarySearch(array1, "bar");
System.out.println(index);
}
}
```

What is the result?

- A. 1
- B. 2
- C. Compilation fails
- D. An exception is thrown at runtime

Answer:

Explanation: Section: (none) Explanation

The code compiles fine. java.lang.NullPointerException

because only one element of list is initialized : element [0] elements [1] and [2] equals null

alte Begründung:

An exception is thrown at runtime due to data type comparison mismatch:

Exception in thread "main" java.lang.ClassCastException: java.lang.String cannot be cast to java.lang.Integer

at java.lang.Integer.compareTo(Integer.java:52)

at java.util.Arrays.binarySearch0(Arrays.java:1481) at java.util.Arrays.binarySearch(Arrays.java:1423) at searchtext.SearchText.main(SearchText.java:22)

Note:binarySearch

public static int binarySearch(char[] a,

char key)Searches the specified array of chars for the specified value using the binary search algorithm. The array mustbe sorted (as by the sort method, above)

prior to making this call. If it is not sorted, the results are undefined. Ifthe array contains multiple elements with the specified value, there is no guarantee which one will be found.

Parameters:

a - the array to be searched.

key - the value to be searched for. Returns:

Indexof the search key, if it is contained in the list; otherwise, -(insertion point) - 1). The insertionpoint is defined as the point at which the key would be inserted into the list: the index of the first elementgreater than the key, or list.size(), if all elements in the list are less than the specified key. Note that thisguarantees that the return value will be >= 0 if and only if the key is found.

NEW QUESTION 11

Given the code fragment:

```
public class DisplaValues {

public void printNums (int [] nums){
for (int number: nums) {

System.err.println(number);

}
}
}
```

Assume the method printNums is passed a valid array containing data. Why is this method not producingoutput on the console?

- A. There is a compilation error.
- B. There is a runtime exception.
- C. The variable number is not initialized.
- D. Standard error is mapped to another destination.

Answer: D

Explanation: The code compiles fine. The code runs fine.

The errorstream can be redirected. Note:

System.out.println -> Sends the output to a standard output stream. Generally monitor. System.err.println -> Sends the output to a standard error stream.

Generally monitor. err is the "standard" erroroutput stream. This stream is already open and ready to accept output data.

Typically this stream corresponds to display output or another output destination specified by the hostenvironment or user. By convention, this output stream is used to display error

messages or other informationthat should come to the immediate attention of a user even if the principal output stream, the value of thevariable out, has been redirected to a file or other destination that is typically not continuously monitored.

Reference:java.lang.System

NEW QUESTION 12

Given:

```
public class SuperThread extends Thread {  
    public void run(String name) {  
        System.out.print("Thread");  
    }  
}
```

```
    public void run(Runnable r) {  
        r = new Runnable() {  
            public void run() {  
                System.out.print("Runnable");  
            }  
        };  
    }  
}
```

```
    public static void main(String[] args) {  
        Thread t = new SuperThread();  
        start();  
    }  
}
```

Which two are true?

- A. Thread is printed
- B. Runnable is printed
- C. No output is produced
- D. No new threads of execution are started within the main method
- E. One new thread of execution is started within the main method
- F. Two new threads of exclusion are started within the main method

Answer: CD

NEW QUESTION 16

Given:

```
public class CowArray extends Thread {
    static List<Integer> myList = new CopyOnWriteArrayList<Integer>();

    public static void main(String[] args) {
        myList.add(11);
        myList.add(22);
        myList.add(33);
        myList.add(44);

        new CowArray().start();

        for (Integer i: myList) {
            try { Thread.sleep(1000); }
            catch (Exception e) { System.out.print("e1 "); }

            System.out.print( " " + i );
        }

        public void run() {
            try { Thread.sleep(500); }
            catch (Exception e) { System.out.print("e2 "); }

            myList.add(77);
            System.out.print("size: " + myList.size() + ", elements:");
        }
    }
}
```

What is the most likely result?

- A. size: 4, elements: 11 22 33 44
- B. size: 5, elements: 11 22 33 44
- C. size: 4, elements: 11 22 33 44 77
- D. size: 5, elements: 11 22 33 44 77
- E. a ConcurrentModification Exception is thrown

Answer: B

NEW QUESTION 18

ITEM Table

* ID, INTEGER: PK

* DESCRIP, VARCHAR(100)

* PRICE, REAL

* QUALITY, INTEGER

And given the code fragment (assuming that the SQL query is valid):

```
try {  
    String query = "SELECT * FROM Item WHERE ID=110";  
    Statement stmt = conn.createStatement();  
    ResultSet rs = stmt.executeQuery(query);  
    while (rs.next ()) {  
        System.out.println("ID: " + rs.getInt("Id"));  
        System.out.println("Description: " + rs.getString("Descrip"));  
        System.out.println("Price: " + rs.getDouble("Price"));  
        System.out.println("Quantity: " + rs.getInt("Quantity"));  
    }  
} catch (SQLException se) {  
    System.out.println( "Error" );  
}
```

What is the result of compiling and executing this code?

- A. An exception is thrown at runtime
- B. Compile fails
- C. The code prints Error
- D. The code prints information about Item 110

Answer: C

Explanation: Tricky:

Compiles successfully ! Not B !

D is correct, if Column Quantity instead of Quality

Table Item Column Quality --- System.out.println("Quantity: " + rs.getInt("Quantity")); wenn jedoch so gewollt: die Zeile gibt Error aus (die anderen funktionieren) !!!

The connection conn is not defined. The code will not compile.

NEW QUESTION 19

Select four examples that initialize a NumberFormat reference using a factory.

- A. NumberFormat nf1 = new DecimalFormat();
- B. NumberFormat nf2 = new DecimalFormat("0.00") ;
- C. NumberFormat nf3 = NumberFormat.getInstance();
- D. NumberFormat nf4 = NumberFormat.getIntegerInstance();
- E. NumberFormat nf5 = DecimalFormat.getNumberInstance ();
- F. NumberFormat nf6 = NumberFormat.getCurrencyInstance () ;

Answer: CDEF

Explanation: getInstance

public static final NumberFormat getInstance()

Returns the default number format for the current default locale. The default format is one of the styles

provided by the other factory methods: getNumberInstance(E), getIntegerInstance(D), getCurrencyInstance(F)

or getPercentInstance. Exactly which one is locale dependant.

C: To obtain a NumberFormat for a specific locale, including the default locale, call one of NumberFormat's factory methods, such as getInstance().

E: To obtain standard formats for a given locale, use the factory methods on NumberFormat such as getNumberInstance. These factories will return the most appropriate sub-class of NumberFormat for a given locale.

F: To obtain standard formats for a given locale, use the factory methods on NumberFormat such as getInstance or getCurrencyInstance.

Reference: java.text Class NumberFormat

NEW QUESTION 21

Given:

```
import java.util.Map;
import java.util.Set;
import java.util.TreeMap;

public class MapClass {
    public static void main(String[] args) {
        Map <String, String> partList = new TreeMap<>();
        partList.put("P002", "Large Widget");
        partList.put("P001", "Widget");
        partList.put("P002", "X-Large Widget");

        Set<String> keys = partList.keySet();

        for (String key:keys) {
            System.out.println(key + " " + partList.get(key));
        }
    }
}
```

What is the result?

- A. p001 Widget p002 X-Large Widget
- B. p002 Large Widget p001 Widget
- C. p002 X-large Widget p001 Widget
- D. p001 Widget p002 Large Widget
- E. compilation fails

Answer: A

Explanation: Compiles fine. Output is: P001 Widget
P002 X-Large Widget

Line: partList.put("P002", "X-Large Widget"); >> overwrites >> line:partList.put("P002", "Large Widget");
put

V put(K key, V value)

Associates the specified value with the specified key in this map (optional operation). If the map previously contained a mapping for the key, the old value is replaced by the specified value. (A map m is said to contain a mapping for a key k if and only if m.containsKey(k) would return true.)

Parameters:

key - key with which the specified value is to be associated value - value to be associated with the specified key

Returns the previous value associated with key, or null if there was no mapping for key. (A null return can also indicate that the map previously associated null with key, if the implementation supports null values.)

NEW QUESTION 22

Given:

```
class InvalidAgeException extends IllegalArgumentException { }

public class Tracker {

    void verify (int age) throws IllegalArgumentException {
        if (age < 12)
            throw new InvalidAgeException ();
        if (age >= 12 && age <= 60)
            System.out.print("General category");
        else
            System.out.print("Senior citizen category");
    }

    public static void main(String[] args) {
        int age = Integer.parseInt(args[1]);
        try {
            new Tracker().verify(age);
        }
        catch (Exception e) {
            System.out.print(e.getClass());
        }
    }
}
```

And the command-line invocation: Java Tracker 12 11
 What is the result?

- A. General category
- B. class InvalidAgeException
- C. class java.lang.IllegalArgumentException
- D. class java.lang.RuntimeException

Answer: B

Explanation: The second argument 11 makes the program to throw an InvalidAgeException due to the line:
 if (age < 12)
 throw new InvalidAgeException ();

NEW QUESTION 24

To provide meaningful output for: System.out.print(new Item ());
 A method with which signature should be added to the Item class?

- A. public String asString()
- B. public Object asString()
- C. public Item asString()
- D. public String toString()
- E. public object toString()
- F. public Item toString()

Answer: D

Explanation: Implementing toString method in java is done by overriding the Object's toString method. The javatoString() method is used when we need a string representation of an object. It is defined in Object class. This method can be overridden to customize the String representation of the Object.

Note:

Below is an example shown of Overriding the default Object toString() method. The toString() method must be descriptive and should generally cover all the contents of the object.

```
class PointCoordinates { private int x, y;
    public PointCoordinates(int x, int y) { this.x = x;
    this.y = y;
    }
```

```
public int getX() { return x;
}
public int getY() { return y;
}
// Custom toString() Method. public String toString() {
return "X=" + x + " " + "Y=" + y;
}}
```

NEW QUESTION 25

Given:

```
interface Rideable {
    public String ride() { return "riding "; }
}
class Horse implements Rideable {
    public String ride() { return "cantering "; }
}
class Icelandic extends Horse implements Rideable {
    public String ride() { return "tolting "; }
}
class Test {
    public static void main(String[] args) {
        Rideable r1 = new Icelandic();
        Rideable r2 = new Horse();
        Horse h1 = new Icelandic();
        System.out.println(r1.ride() + r2.ride() + h1.ride());
    }
}
```

- A. riding riding tolting
- B. riding riding cantering
- C. tolting cantering tolting
- D. tolting cantering cantering
- E. Compilation fails.
- F. An exception is thrown at runtime.

Answer: E

Explanation: The compilation fails at: interface Rideable {
public String ride() { return "riding ";}
}

Error due to: interface methods cannot have body.

NEW QUESTION 26

For which three objects must a vendor provide implementations in its JDBC driver?

- A. Time
- B. Date
- C. Statement
- D. ResultSet
- E. Connection
- F. SQLException
- G. DriverManager

Answer: CDE

Explanation: All JDBC drivers implement the four important JDBC classes: Driver, Connection, Statement, and ResultSet.

NEW QUESTION 31

- Index.htm
- Service.html
- Logo.gif
- Title.jpg

And the code fragment:

```
public class SearchApp extends SimpleFileVisitor<Path> {
    private final PathMatcher matcher;

    public SearchApp() {
        matcher = FileSystems.getDefault().getPathMatcher( "glob:* .htm,html,xml" );
    }
    void search(Path file){
        Path name = file.getFileName();
        if (name != null && matcher.matches(name)) {
            System.out.println(name);
        }
    }
    public FileVisitResult visitFile (Path file, BasicFileAttributes attrs) {
        search(file);
        return FileVisitResult.CONTINUE;
    }
    public static void main(String[] args) throws IOException {
        Files.walkFileTree(Paths.get("doc"), new SearchApp());
    }
}
```

What is the result, if doc is present in the current directory?

- A. No output is produced.
- B. index.htm
- C. index.htm userguide.txt logo.gif
- D. index.htm service.html userguide.txt logo.gif

Answer: A

Explanation: The Glob search expression is defined through "glob:* .htm, html, xml" The correct answer is A
The glob is trying to match all the string. The correct way is
glob:*.{htm,html,xml}
and then would be found: Index.htm
Service.html

NEW QUESTION 33

Given the incomplete pseudo-code for a fork/join framework application:

```
submit(Data) {  
  
if(Data.size < SMALL_ENOUGH) {  
  
    _____(Data); // line x  
  
}  
  
else {  
  
List<Data> x = _____(Data); // line Y  
  
for(Data d: x  
  
    _____(d); // line z  
  
}  
  
}
```

And given the missing methods: Process, submit, and splitInHalf
Which three insertions properly complete the pseudo-code?

- A. Insert submit at line X.
- B. Insert splitInHalf at line X.
- C. Insert process at line X.
- D. Insert process at line Y.
- E. Insert splitInHalf at line Y.
- F. Insert process at line Z.
- G. Insert submit at line Z.

Answer: CEG

Explanation: C: If data is small enough then process it. Line X
E: If data is not small enough then split it half. Line Y
G: After the data has been split (line Y) then recursively submit the splitted data (Line z).

NEW QUESTION 37

Given:

```
class Plant {  
    abstract String growthDirection();  
}  
  
class Embryophyta extends Plant {  
    String growthDirection() { return "Up " }  
}  
  
public class Garden {  
    public static void main(String[] args) {  
        Embryophyta e = new Embryophyta();  
        Embryophyta c = new Carrot();  
        System.out.print(e.growthDirection() + growthDirection());  
    }  
}
```

What is the result?

- A. Up Down
- B. Up Up
- C. Up null
- D. Compilation fails
- E. An exception is thrown at runtime

Answer: D

Explanation: Exception in thread "main" java.lang.ExceptionInInitializerError at garden.Garden.main Caused by: java.lang.RuntimeException: Uncompilable source code - garden.Plant is not abstract and doesnot override abstract method growthDirection() in garden.Plant

NEW QUESTION 39

Given:

```
interface Car {  
    public void start();  
}  
  
class BasicCar implements Car {  
    public void start() {}  
}  
  
public class SuperCar {  
    Car c = new BasicCar ();  
    public void start() {  
        c.start();  
    }  
}
```

Which three are true?

- A. BasicCar uses composition.
- B. SuperCar uses composition.
- C. BasicCar is-a Car.
- D. SuperCar is-a Car.
- E. SuperCar takes advantage of polymorphism
- F. BasicCar has-a Car

Answer: BCE

Explanation: B: The relationship modeled by composition is often referred to as the "has-a" relationship. Here SuperCarhas-a Car.

C:The relationship modeled by inheritance is often referred to as the "is-a" relationship. Modeling an is-arelationship is called inheritance because the subclass inherits the interface and, by default, theimplementation of the superclass. Inheritance of interface guarantees that a subclass can accept all the samemessages as its superclass. A subclass object can, in fact, be used anywhere a superclass object is called for.E:The polymorphic method call allows one type to express its distinction from another, similar type, as long asthey're both derived from the same base type. This distinction is expressed through differences in behavior ofthe methods that you can call through the base class.

NEW QUESTION 41

Given:

```
final class FinalShow { // Line 1
final String location; // Line 2
FinalShow(final String loc) { // Line 3
location = loc; // Line 4
} // Line 5
```

```
FinalShow(String loc, String title) { // Line 6
location = loc; // Line 7
loc = "unknown"; // Line 8
} // Line 9
} // Line 10
```

What is the result?

- A. Compilation succeeds.
- B. Compilation fails due to an error on line 1.
- C. Compilation fails due to an error on line 2.
- D. Compilation fails due to an error on line 3.
- E. Compilation fails due to an error on line 4.
- F. Compilation fails due to an error on line 8.

Answer: A

NEW QUESTION 44

Given:

```
import java.util.concurrent.atomic.AtomicInteger;

public class Incrementor {

    public static void main(String[] args) {
        AtomicInteger[] var = new AtomicInteger[5];
        for (int i = 0; i < 5; i++) {
            var[i] = new AtomicInteger();
            System.out.print(var[i] + " ");
        }
        System.out.println();

        for (int i = 0; i < var.length; i++) {
            var[i].incrementAndGet();
            if (i == 2)
                var[i].compareAndSet(2,4);

            System.out.print(var[i] + " ");
        }
    }
}
```

What is the result?

- A. 1 1 1 1 1
- B. 1 2 3 4 5
- C. 0 1 2 3 4
- D. 0 1 4 3 4

Answer: A

Explanation: first for-loop set 0 0 0 0 0
second for-loop increments each to 1 1 1 1 1
if condition is not given

NEW QUESTION 48

Given:

```
class Counter extends Thread {
    int i = 10;
    public synchronized void display ( Counter obj) {
        try {
            Thread.sleep (5) ;
            obj.increment (this) ;
            System.out.println (i) ;
        } catch (InterruptedException ex) { }
    }
    public synchronized void increment (Counter obj) {
        i++;
    }
}
```

```
public class Test {
    public static void main (String[] args) {
        final Counter obj1 = new Counter() ;
        final Counter obj2 = new Counter() ;
        new Thread( new Runnable() {
            public void run() {
                obj1.display(obj2) ;
            }
        }).start() ;
        new Thread( new Runnable() {
            public void run() {
                obj2.display(obj1) ;
            }
        }).start() ;
    }
}
```

From what threading problem does the program suffer?

- A. deadlock
- B. livelock
- C. starvation
- D. race condition

Answer: B

Explanation: A thread often acts in response to the action of another thread. If the other thread's action is also a response to the action of another thread, then livelock may result. As with deadlock, livelocked threads are unable to make further progress.

However, the threads are not blocked -- they are simply too busy responding to each other to resume work.

This is comparable to two people attempting to pass each other in a corridor: Alphonse moves to his left to let Gaston pass, while Gaston moves to his right to let Alphonse pass. Seeing that they are still blocking each other, Alphonse moves to his right, while Gaston moves to his left. They're still blocking each other, so.

NEW QUESTION 53

Given the code fragment: DateFormat df;

Which statement defines a new Dateformat object that displays the default date format for the UK Locale?

- A. df = DateFormat.getDateInstance (DateFormat.DEFAULT, Locale (UK));
- B. df = DateFormat.getDateInstance (DateFormat.DEFAULT, UK);
- C. df = DateFormat.getDateInstance (DateFormat.DEFAULT, Locale.UK);
- D. df = new DateFormat.getDateInstance (DateFormat.DEFAULT, Locale.UK);
- E. df = new DateFormat.getDateInstance (DateFormat.DEFAULT, Locale (UK));

Answer: C

Explanation: The UK locale is constructed withLocale.UK.

To format a date for a different Locale, specify it in the call to getDateInstance(). DateFormat df =

DateFormat.getDateInstance(DateFormat.LONG, Locale.FRANCE); Note: getDateInstance(int style, Locale aLocale)

Gets the date formatter with the given formatting style for the given locale. Reference:Class DateFormat

NEW QUESTION 58

Given:

```
class Erupt implements Runnable {
    public void run() {
        System.out.print(Thread.currentThread().getName());
    }
}

public class Yellowstone {
    static Erupt e = new Erupt();
    Yellowstone() { new Thread( e, "const" ).start(); } // line A

    public static void main(String[] args) {
        new Yellowstone();
        new Faithful().go();
    }
    static class Faithful {
        void go() { new Thread( e, "inner" ).start(); } // line B
    }
}
```

What is the result?

- A. Both const and inner will be in the output.
- B. Only const will be in the output.
- C. Compilation fails due to an error on line A.
- D. Compilation fails due to an error on line B.
- E. An Exception is thrown at runtime.

Answer: A

Explanation: The code compiles fine.

Note:The Runnable interface should be implemented by any class whose instances are intended to beexecuted by a thread. The class must define a method of no arguments called run.

This interface is designed to provide a common protocol for objects that wish to execute code while they areactive. For example, Runnable is implemented by class Thread. Being active simply means that a thread hasbeen started and has not yet been stopped.

In addition, Runnable provides the means for a class to be active while not subclassing Thread. Aclass that implements Runnable can run without subclassing Thread by instantiating a Thread instance andpassing itself in as the target. In most cases, the Runnable interface should be used if you are only planning tooverride the run() method and no other Thread methods. This is important because classes should not besubclassed unless the programmer intends on modifying or enhancing the fundamental behavior of the class.

Note 2:start()

Causes this thread to begin execution; the Java Virtual Machine calls the run method of this thread.

Reference: java.lang
Interface Runnable

NEW QUESTION 59

Which two statements are true about RowSet subinterfaces?

- A. A JdbcRowSet object provides a JavaBean view of a result set.
- B. A CachedRowSet provides a connected view of the database.
- C. A FilteredRowSet object filter can be modified at any time.
- D. A WebRowSet returns JSON-formatted data.

Answer: AC

Explanation: A: a JdbcRowSet object can be one of the Beans that a tool makes available for composing an application.

Because a JdbcRowSet is a connected RowSet, that is, it continually maintains its connection to a database using a JDBC technology-enabled driver, it also effectively makes the driver a JavaBeans component.

C: The FilteredRowSet range criterion can be modified by applying a new Predicate object to the FilteredRowSet instance at any time. This is possible if no additional references to the FilteredRowSet object are detected. A new filter has an immediate effect on criterion enforcement within the FilteredRowSet object, and all subsequent views and updates will be subject to similar enforcement.

Reference: javax.sql Interface RowSet

NEW QUESTION 64

Given:

```
interface Event {  
    String type = "Event";  
    public void details();  
}  
  
class Quiz {  
    static String type = "Quiz";  
}  
  
public class PracticeQuiz extends Quiz implements Event {  
    public void details() {  
        System.out.print(type);  
    }  
  
    public static void main(String[] args) {  
        new PracticeQuiz().details();  
        System.out.print(" " + type);  
    }  
}
```

What is the result?

- A. Event Quiz
- B. Event Event
- C. Quiz Quiz
- D. Quiz Event
- E. Compilation fails

Answer: E

NEW QUESTION 65

Given:

```
import java.util.ArrayList;
import java.util.List;

interface Glommer { }
interface Plinkable { }

public class Flimmer implements Plinkable {
    List<Tagget> t = new ArrayList<Tagget>();
}

class Flommer extends Flimmer { }

class Tagget {
    void doStuff() {
        String s = "yo";
    }
}
```

Which three statements concerning the OO concepts "is-a" and "has-a" are true?

- A. Flimmer is-a Plinkable
- B. Flommer has-a Tagget
- C. Flommer is-a Glommer
- D. Tagget has-a String
- E. Flommer is-a Plinkable
- F. Flimmer is-a Flommer
- G. Tagget is-a Plinkable

Answer: ABE

Explanation: A: Flimmer implements Plinkable. Flimmer is-a plinkable.

D:The relationship modeled by composition is often referred to as the "has-a" relationship. HereTaggethasaString.

F: Flommer extends Flimmer

So there is an "is-a relationship between Flommer and Flimmer .

Note: Thehas-a relationship has anencapsulation feature (like private or protected modifier used before eachmember field or method).

NEW QUESTION 68

Which four are true about enums?

- A. An enum is typesafe.
- B. An enum cannot have public methods or fields.
- C. An enum can declare a private constructor.
- D. All enums implicitly implement Comparable.
- E. An enum can subclass another enum.
- F. An enum can implement an interface.

Answer: ACDF

Explanation: C: The constructor for an enum type must be package-private or private access. Reference: Java Tutorials,Enum Types

NEW QUESTION 71

Which is a key aspect of composition?

- A. Using inheritance
- B. Method delegation
- C. Creating abstract classes
- D. Implementing the composite interface

Answer: B

Explanation: In the composition approach, the subclass becomes the "front-end class," and the superclass becomes the "back-end class." With inheritance, a subclass automatically inherits an implementation of any non-private superclass method that it doesn't override. With composition, by contrast, the front-end class must explicitly invoke a corresponding method in the back-end class from its own implementation of the method. This explicit call is sometimes called "forwarding" or "delegating" the method invocation to the back-end object. Note: Composition means the same as:

* contains

* is part of

Note 2: As you progress in an object-oriented design, you will likely encounter objects in the problem domain that contain other objects. In this situation you will be drawn to modeling a similar arrangement in the design of your solution. In an object-oriented design of a Java program, the way in which you model objects that contain other objects is with composition, the act of composing a class out of references to other objects. With composition, references to the constituent objects become fields of the containing object. To use composition in Java, you use instance variables of one object to hold references to other objects.

NEW QUESTION 76

Given:

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class NameList {
    public static void main(String[] args) {
        List<String> nameList = new ArrayList<>(3);

        nameList.add("John Adams");
        nameList.add("George Washington");
        nameList.add("Thomas Jefferson");

        Collections.sort(nameList);
        for (String name : nameList) {
            System.out.println(name);
        }
    }
}
```

What is the result?

- A. John Adams George Washington Thomas Jefferson
- B. George Washington John Adams Thomas Jefferson
- C. Thomas Jefferson John Adams George Washington
- D. An exception is thrown at runtime
- E. Compilation fails

Answer: B

Explanation: The program compiles and runs fine.

At runtime the NameList is built and then sorted by natural Order (String >> alphabetically).

NEW QUESTION 80

Given:

What is the result?

- A. fast slow
- B. fast goes
- C. goes goes
- D. fast fast
- E. fast followed by an exception
- F. Compilation fails

Answer: F

Explanation: Line:Vehicle v = new Sportscar(); causes compilation failure:

error: cannot find symbol Vehicle v = new Sportscar(); symbol: class Sportscar location: class VehicleTest

NEW QUESTION 81

Given:

```
import java.util.ArrayList;
import java.util.List;

interface Glommer { }
interface Plinkable { }

public class Flimmer implements Plinkable {
    List<Tagget> t = new ArrayList<Tagget>();
}

class Flommer extends Flimmer {
    String s = "hey";
}

class Tagget implements Glommer {
    void doStuff() {
        String s = "yo";
    }
}
```

Which two statements concerning the OO concepts "IS-A" and "HAS-A" are true?

- A. Flimmer is-a Glommer.
- B. Flommer has-a String.
- C. Tagget has-a Glommer.
- D. Flimmer is-a ArrayList.
- E. Tagget has-a doStuff()
- F. Tagget is-a Glommer.

Answer: BF

Explanation: B: The relationship modeled by composition is often referred to as the "has-a" relationship. Here Flommer hasaString.

E: The has-a relationship has an encapsulation feature (like private or protected modifier used before eachmember field or method).

Here Tagget has-a method doStuff() F: Tagget implements Glommer. Tagget is-a Glommer.

Note: The has-a relationship has an encapsulation feature (like private or protected modifier used before eachmember field or method).

NEW QUESTION 83

Given these facts about Java classes in an application:

- Class X is-a Class SuperX.
- Class SuperX has-a public reference to a Class Z.
- Class Y invokes public methods in Class Util.
- Class X uses public variables in Class Util. Which three statements are true?

- A. Class X has-a Class Z.
- B. Class Util has weak encapsulation.
- C. Class Y demonstrates high cohesion.
- D. Class X is loosely coupled to Class Util.
- E. Class SuperX's level of cohesion CANNOT be determined

Answer: BDE

Explanation: B: Has class Util has both public methods and variables, it is an example of weak encapsulation.

Note: Inheritance is also sometimes said to provide "weak encapsulation," because if you have code that directly uses a subclass, such as Apple, that code can be broken by changes to a superclass, such as Fruit.

One of the ways to look at inheritance is that it allows subclass code to reuse superclass code. For example, if Apple doesn't override a method defined in its superclass Fruit, Apple is in a sense reusing Fruit's implementation of the method. But Apple only "weakly encapsulates" the Fruit code it is reusing, because changes to Fruit's interface can break code that directly uses Apple.

D:

Note: Tight coupling is when a group of classes are highly dependent on one another.

This scenario arises when a class assumes too many responsibilities, or when one concern is spread over many classes rather than having its own class.

Loose coupling is achieved by means of a design that promotes single-responsibility and separation of concerns.

A loosely-coupled class can be consumed and tested independently of other (concrete) classes.

Interfaces are a powerful tool to use for decoupling. Classes can communicate through interfaces rather than other concrete classes, and any class can be on the other end of that communication simply by implementing the interface.

E: Not enough information regarding SuperX' to determine the level of cohesion.

NEW QUESTION 84

Given:
 StringBuffer b = new StringBuffer("3"); System.out.print(5+4+b+2+1);
 What is the result?

- A. 54321
- B. 9321
- C. 5433
- D. 933
- E. Output is Similar to: 9java.lang.StringBuffer@100490121.
- F. Compilation fails.

Answer: F

Explanation: The code will not compile.
 The print function cannot handle the mixture of integers and strings.
 Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - Erroneous tree type

NEW QUESTION 85

Given:

```
interface Books {

    //insert code here

}
```

Which fragment, inserted in the Books interface, enables the code to compile?

- A. public abstract String type; public abstract String getType();
- B. public static String type; public abstract String getType();
- C. public String type = "Fiction"; public static String getType();
- D. public String type = "Fiction"; public abstract String getType();

Answer: D

NEW QUESTION 87

Given:

```
public class Print01 {
    public static void main(String[] args) {
        double price = 24.99;
        int quantity = 2;
        String color = "Blue";
        // insert code here. Line ***
    }
}
```

Which two statements, inserted independently at line ***, enable the program to produce the following output:

We have 002 Blue pants that cost \$24.99.

- A. System.out.printf("We have %03d %s pants that cost \$%3.2f.\n",quantity, color, price);
- B. System.out.printf("We have\$03d\$s pants that cost \$\$3.2f.\n",quantity, color, price);
- C. String out = String.format ("We have %03d %s pants that cost \$%3.2f.\n",quantity, color,price);System.out.println(out);
- D. String out = System.out.format("We have %03d %s pants that cost \$%3.2f.",quantity, color, price);System.out.println(out);
- E. System.out.format("We have %s%spants that cost \$%s.\n",quantity, color, price);

Answer: AC

NEW QUESTION 88

Given the code fragment:

```
try {
    conn.setAutoCommit(false);
    stmt.executeUpdate("insert into employees values(1,'Sam')");
    Savepoint save1 = conn.setSavepoint("point1");
    stmt.executeUpdate("insert into employees values(2,'Jane')");
    conn.rollback();
    stmt.executeUpdate("insert into employees values(3,'John')");
    conn.setAutoCommit(true);
    stmt.executeUpdate("insert into employees values(4,'Jack')");
    rs = stmt.executeQuery("select * from employees");
    while (rs.next()) {
        System.out.println(rs.getString(1) + " " + rs.getString(2));
    }
} catch (Exception e) {
    System.out.print(e.getMessage());
}
```

What is the result of the employees table has no records before the code executed?

- A. 1 Sam
- B. 4 Jack
- C. 3 John4 Jack
- D. 1 Sam3 John4 Jack

Answer: C

Explanation: AutoCommit is set to false. The two following statements will be within the same transaction.

stmt.executeUpdate("insert into employees values(1,'Sam')"); stmt.executeUpdate("insert into employees values(2,'Jane')");

These two statements are rolled-back through (the savepoint is ignored! the savepoint must be specified (e.g.

conn.rollback(save1);) in the rollback if you want to rollback to the savepoint): conn.rollback() ;

The next two insert statements are executed fine. Their result will be in the output.

NEW QUESTION 90

Given the two Java classes:

```
class Word {  
  
    private Word(int length) {}  
    protected Word(String w) {}  
  
}  
  
public class Buzzword extends Word {  
    public Buzzword() {  
        // Line ***, add code here  
    }  
  
    public Buzzword(String s) {  
        super(s);  
    }  
}
```

Which two code snippets, added independently at line ***, can make the Buzzword class compile?

- A. this ();
- B. this (100);
- C. this ("Buzzword");
- D. super ();
- E. super (100);
- F. super ("Buzzword");

Answer: CF

NEW QUESTION 92

Given:

```
import java.util.ArrayDeque;
import java.util.Deque;

public class Counter {
    public static void main(String[] args) {
        Deque<String> deq = new ArrayDeque<String>(2);

        deq.addFirst("one");
        deq.addFirst("two");
        deq.addFirst("three"); // Line 9

        System.out.print(deq.pollLast());
        System.out.print(deq.pollLast());
        System.out.print(deq.pollLast()); // Line 12
    }
}
```

What is the result?

- A. An exception is thrown at runtime on line 9.
- B. An exception is thrown at runtime on line 12
- C. onetwonull
- D. onetwothree
- E. twoonenull
- F. threetwoone

Answer: D

Explanation: addFirst

void addFirst(E e)

Inserts the specified element at the front of this deque if it is possible to do so immediately without violating capacity restrictions. When using a capacity-restricted deque, it is generally preferable to use method offerFirst (E).

pollLast

E pollLast()

Retrieves and removes the last element of this deque, or returns null if this deque is empty. Returns: the tail of this deque, or null if this deque is empty

NEW QUESTION 96

Given the code format: SimpleDateFormat sdf;

Which code statements will display the full text month name?

- A. sdf = new SimpleDateFormat ("mm", Locale.UK); System.out.println("Result:", sdf.format(new date()));
- B. sdf = new SimpleDateFormat ("MM", Locale.UK); System.out.println("Result:", sdf.format(new date()));
- C. sdf = new SimpleDateFormat ("MMM", Locale.UK); System.out.println("Result:", sdf.format(new date()));
- D. sdf = new SimpleDateFormat ("MMMM", Locale.UK); System.out.println("Result:", sdf.format(new date()));

Answer: D

Explanation: Typical output would be Current Month in M format: 2

Current Month in MM format: 02 Current Month in MMM format: Feb

Current Month in MMMM format: February

NEW QUESTION 97

Given the directory structure that contains three directories: company, Salesdat, and Finance:

Company

- Salesdat

* Target.dat

- Finance

* Salary.dat

* Annual.dat

And the code fragment:

```
public class SearchApp extends SimpleFileVisitor<Path> {
    private final PathMatcher matcher;
    SearchApp() {
        matcher = FileSystems.getDefault().getPathMatcher("glob:*dat");
    }
    void find(Path file){
        Path name = file.getFileName();
        if (name != null && matcher.matches(name)){
            System.out.println(name);
        }
    }
    public FileVisitResult visitFile (Path file, BasicFileAttributes atr){
        find(file);
        return FileVisitResult.CONTINUE;
    }
    public static void main(String[] args) throws IOException {
        SearchApp obj = new SearchApp();
        Files.walkFileTree(Paths.get("c:/_Workspace/OCP/src/examrest/Company/"), obj);
    }
}
```

If Company is the current directory, what is the result?

- A. Prints only Annual.dat
- B. Prints only Salesdat, Annual.dat
- C. Prints only Annual.dat, Salary.dat, Target.dat
- D. Prints at least Salesdat, Annual.dat, Salary.dat, Target.dat

Answer: A

Explanation: IF !! return FileVisitResult.CONTINUE;

The pattern *dat will match the directory name Salesdat and it will also match the file Annual.dat.

It will not be matched to Target.dat which is in a subdirectory.

NEW QUESTION 100

The two methods of code reuse that aggregate the features located in multiple classes are _____ ?

- A. Inheritance
- B. Copy and Paste
- C. Composition
- D. Refactoring
- E. Virtual Method Invocation

Answer: AC

Explanation: A: Inheritance is a way of reusing code and building bigger more functional objects from a basic object.

The original little object, the parent, is called the super-class. The more functional object that inherits from it is called the sub-class .

C: When your goal is code reuse, composition provides an approach that yields easier-to- change code.

NEW QUESTION 105

Which method would you supply to a class implementing the Callable interface?

- A. callable ()
- B. executable ()
- C. call ()
- D. run ()
- E. start ()

Answer: C

Explanation: public interface Callable<V>

A task that returns a result and may throw an exception. Implementors define a single method with no arguments called call.

Note:

Interface Callable<V> Type Parameters:

V - the result type of method call

The Callable interface is similar to Runnable, in that both are designed for classes whose instances are potentially executed by another thread. A Runnable, however, does not return a result and cannot throw a checked exception.

The Executors class contains utility methods to convert from other common forms to Callable classes.

Reference: java.util.concurrent

NEW QUESTION 106

Given:

```
import java.io.File;
import java.nio.file.Path;

public class Test12 {

    static String displayDetails(String path, int location) {
        Path p = new File(path).toPath();
        String name = p.getName(location).toString();
        return name;
    }

    public static void main(String[] args) {
        String path = "project//doc//index.html";
        String result = displayDetails(path,2);
        System.out.print(result);
    }
}
```

What is the result?

- A. doc
- B. index.html
- C. an IllegalArgumentException is thrown at runtime.
- D. An InvalidPthException is thrown at runtime.
- E. Compilation fails.

Answer: B

Explanation: p.getName(int location) = returns path' name element by index/location (starts with 0) Example:
path = "project//doc//index.html" p.getName(0) = project p.getName(1) = doc p.getName(2) = index.html

NEW QUESTION 110

Which code fragment demonstrates the proper way to handle JDBC resources?

- A. try {ResultSet rs = stmt.executeQuery (query); statement stmt = con.createStatement(); while (rs.next()) { /* . . . */} catch (SQLException e) {}
- B. try {Statement stmt = con.createStatement(); ResultSet rs = stmt.executeQuery (query); while (rs.next()) { /* . . . */} catch (SQLException e) {}
- C. try {Statement stmt = con.createStatement();ResultSet rs = stmt.executeQuery (query); while (rs.next()) { /* . . . */} finally { rs.close();stmt.close();}
- D. try {ResultSet rs = stmt.executeQuery (query); Statement stmt = con.createStatement(); while (rs.next()) { /* . . . */} finally { rs.close();stmt.close();}

Answer: C

NEW QUESTION 113

Given:

```
public class Dog {  
protected String bark() {return "woof "; }  
}  
public class Beagle extends Dog {  
private String bark() { return "arf "; }  
}  
public class TestDog {  
public static void main(String[] args) {  
Dog[] dogs = {new Dog(), new Beagle()};  
for(Dog d: dogs)  
System.out.print(d.bark());  
}  
}
```

What is the result?

- A. woof arf
- B. woof woof
- C. arf arf
- D. A RuntimeException is generated
- E. The code fails to compile

Answer: E

Explanation: class Dog {
protected String bark()
public class Beagle extends Dog { private String bark()
Cannot reduce the visibility of the inherited method from Dog

NEW QUESTION 117

Given:

```
public abstract class Account {  
    abstract void deposit (double amt);  
    public abstract Boolean withdraw (double amt);  
}  
  
public class CheckingAccount extends Account {  
  
}
```

What two changes, made independently, will enable the code to compile?

- A. Change the signature of Account to: public class Account.
- B. Change the signature of CheckingAccount to: public abstract CheckingAccount
- C. Implement private methods for deposit and withdraw in CheckingAccount.
- D. Implement public methods for deposit and withdraw in CheckingAccount.
- E. Change Signature of checkingAccount to: CheckingAccount implements Account.
- F. Make Account an interface.

Answer: BD

Explanation: Compiler say:

- Der Typ CheckingAccount muss die übernommene abstrakte Methode Account.deposit(double) implementieren
 - Der Typ CheckingAccount muss die übernommene abstrakte Methode Account.withdraw(double) implementieren
- ODER
Typ CheckingAccount als abstract definieren

NEW QUESTION 118

Which two are valid initialization statements?

- A. Map<String, String> m = new SortedMap<String, String>();
- B. Collection m = new TreeMap<Object, Object>();
- C. HashMap<Object, Object> m = new SortedMap<Object, Object>();
- D. SortedMap<Object, Object> m = new TreeMap<Object, Object> ();
- E. Hashtable m= new HashMap();
- F. Map<List, ArrayList> m = new Hashtable<List, ArrayList>();

Answer: DF

NEW QUESTION 122

Given that myFile.txt contains:

First
Second
Third

And given:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class ReadFile04 {
    public static void main(String[] args) {
        try (BufferedReader buffln = new BufferedReader(
            new FileReader("D:\\faculty\\myfile.txt"))) {
            String line = "";
            int count = 1;
            buffln.mark(1);
            line = buffln.readLine();
            System.out.println(count + ": " + line);
            line = buffln.readLine();
            count++;
            System.out.println(count + ": " + line);
            buffln.reset();
            line = buffln.readLine();
            count++;
            System.out.println(count + ": " + line);
        } catch (IOException e) {
            System.out.println("IOException");
        }
    }
}
```

What is the result?

- A. 1: First2: Second3: Third
- B. 1: First2: Second3: First
- C. 1: First2: First3: First
- D. IOException
- E. Compilation fails

Answer: B

Explanation: BufferedReader: mark() : Marks the present position in the stream. Subsequent calls to reset() will attempt to reposition the stream to this point.
reset() : Resets the stream to the most recent mark.

!! After last Line is read (readLine()), a trial to reset() throws IOException : Mark invalid

NEW QUESTION 123

The default file system includes a logFiles directory that contains the following files:

Log-Jan 2009 log_Ol_2010 log_Feb2010 log_Feb2011 log_10.2012 log-sum-2012

How many files does the matcher in this fragment match?

PathMatcher matcher = FileSystems.getDefault ().getPathMatcher ("glob: *???_*1?");

- A. One
- B. Two
- C. Three
- D. Four
- E. Five
- F. Six

Answer: B

Explanation: The pattern to match is *??*_1? (regex ".*..._.*1.")

This means at least three characters before the symbol `_`, followed by any amount of characters. The next to last character must be 1. The last character can be any character. The following file names match this pattern:

log_Feb2011

log_10.2012 Trap !! I is not 1 !!

NEW QUESTION 128

Which concept allows generic collections to interoperate with java code that defines collections that use rawtypes?

- A. bytecode manipulation
- B. casting
- C. autoboxing
- D. auto-unboxing
- E. type erasure

Answer: E

Explanation: The type erasure of its leftmost bound, or type Object if no bound was specified. Examples:

type parameters type erasure List<String> List Map.Entry<String,Long> Map.Entry

<T extends Cloneable & Comparable<T>> Cloneable

<T extends Object & Comparable<T>> Object

<T> T[] toArray(T[] a) Object[] toArray(Object[] a)

The type erasure process can be imagined as a translation from generic Java source code back into regular Java code. In reality the compiler is more efficient and translates directly to Java byte code. But the byte code created is equivalent to the non-generic Java code.

NEW QUESTION 131

Given:

```
public class Test {

    public static void main(String[] args) {
        String[] arr = {"SE", "ee", "ME"};
        for(String var : arr) {
            try {
                switch(var) {
                    case "SE":
                        System.out.println("Standard Edition");
                        break;
                    case "EE":
                        System.out.println("Enterprise Edition");
                        break;
                    default: assert false;
                }
            } catch (Exception e) {
                System.out.println(e.getClass());
            }
        }
    }
}
```

And the commands:

```
javac Test.java
java ea Test
```

And the commands:

```
javac Test.java java ea Test
What is the result?
```

- A. Compilation fails
- B. Standard Edition Enterprise Edition Micro Edition
- C. Standard Editionclass java.lang.AssertionError Micro Edition
- D. Standard Edition is printed and an Assertion Error is thrown

Answer: D

Explanation: javac Test.java

will compile the program. As for command line: java ea Test

First the code will produce the output: Standard Edition

See Note below.

The ea option will enable assertions. This will make the following line in the switch statement to be run:

default: assert false;

This will throw an assertion error. This error will be caught. An the class of the assertion error (classjava.lang.AssertionError) will be printed by the following line:

System.out.println(e.getClass());

Note:The java tool launches a Java application. It does this by starting a Java runtime environment, loading aspecified class, and invoking that class's main method. The method declaration must look like the following:

public static void main(String args[]) Paramater ea:

-enableassertions[:<package name>"..." | :<class name>] -ea[:<package name>"..." |

:<class name>]

Enable assertions. Assertions are disabled by default. With no arguments, enableassertions or -ea enablesassertions.

Note 2:

An assertion is a statement in the Java™ programming language that enables you to test your assumptions about your program. Each assertion contains a boolean expression that you believe will be true when the assertion executes. If it is not true, the system will throw an error. public class AssertionError extends Error Thrown to indicate that an assertion has failed. Note 3: The javac command compiles Java source code into Java bytecodes. You then use the Java interpreter - the java command - to interpret the Java bytecodes. Reference: java - the Java application launcher
Reference: java.lang.Class AssertionError

NEW QUESTION 136

Given:

```
interface Writable {  
    void write (String s);  
}
```

```
abstract class Writer implements Writable {  
    // Line ***  
}
```

Which two statements are true about the writer class?

- A. It compiles without any changes.
- B. It compiles if the code void write (String s); is added at line***.
- C. It compiles if the code void write (); is added at line ***.
- D. It compiles if the code void write (string s) { } is added at line ***.
- E. It compiles if the code write () {} is added at line ***.

Answer: A

Explanation: An abstract class does not need to implement the interface methods.

NEW QUESTION 138

An application is waiting for notification of changes to a tmp directory using the following code statements:

```
Path dir = Paths.get("tmp")  
WatchKey key = dir.register (watcher, ENTRY_CREATE, ENTRY_DELETE, ENTRY_MODIFY) ;  
In the tmp directory, the user renames the file testA to testB, Which statement is true?
```

- A. The events received and the order of events are consistent across all platforms.
- B. The events received and the order of events are consistent across all Microsoft Windows versions.
- C. The events received and the order of events are consistent across all UNIX platforms.
- D. The events received and the order of events are platform dependent.

Answer: A

Explanation: Most file system implementations have native support for file change notification. The WatchService API takes advantage of this support where available.

However, when a file system does not support this mechanism, the WatchService will poll the file system, waiting for events.

Note:

WatchKey : When a Watchable entity is registered with a WatchService a key which is a WatchKey is generated. Initially the key is in ready state waiting to be notified of any events on the Watchable entity. Once an event occurs the key goes into signaled state and allows to access the events using its pollEvents method. After processing the poll events the key has to be reset by invoking its reset method. Reference: The Java Tutorials, Watching a Directory for Changes

NEW QUESTION 139

Given the existing destination file, a source file only 1000 bytes long, and the code fragment:

```
public void process (String source, String destination) {  
    try (InputStream fis = new FileInputStream(source);  
        OutputStream fos = new FileOutputStream(destination) ) {  
  
        byte [] buff = new byte[2014];  
        int i;  
        while ((i = fis.read(buff)) != -1) {  
            fos.write(buff,0,i); // line ***  
        }  
    } catch (IOException e) {  
        System.out.println(e.getClass());  
    }  
}
```

What is the result?

- A. Overrides the content of the destination file with the source file content
- B. Appends the content of the source file to the destination file after a new line
- C. Appends the content of the source file to the destination file without a break in the flow
- D. Throws a runtime exception at line***

Answer: A

Explanation: The whole of the FileInputStream will be read (see ** below).

The content of the FileInputStream will overwrite the destination file (see *** below).

*A FileInputStream obtains input bytes from a file in a file system. What files are available depends on the host environment.

FileInputStream is meant for reading streams of raw bytes such as image data. For reading streams of characters, consider using FileReader.

**FileInputStream.read(byte[] b)

Reads up to b.length bytes of data from this input stream into an array of bytes. Parameters:

b - the buffer into which the data is read.

Returns: the total number of bytes read into the buffer, or -1 if there is no more data because the end of the file has been reached.

***FileOutputStream

You can construct a FileOutputStream object by passing a string containing a path name or a File object.

You can also specify whether you want to append the output to an existing file. public FileOutputStream (String path)

public FileOutputStream (String path, boolean append) public FileOutputStream (File file)

public FileOutputStream (File file, boolean append)

With the first and third constructors, if a file by the specified name already exists, the file will be overwritten.

To append to an existing file, pass true to the second or fourth constructor. Reference: Class FileInputStream

Reference: Class FileOutputStream

NEW QUESTION 141

Given:

```

public class AccessTest {
    public static void main(String[] args) {
        Thread t1 = new Thread(new WorkerThread());
        Thread t2 = new Thread(new WorkerThread());
        t1.start(); t2.start; // line1
    }
}

class WorkPool {

    static ArrayList<Integer> list = new ArrayList<>(); // line2
    public static void addItem() { // line3
        list.add(1); // Line4
    }
}

class WorkerThread implements Runnable {
    static Object bar = new Object ();
    public void run() { //line5
        for (int i=0; i<5000;i++) WorkPool.addItem(); // line6
    }
}

```

Which of the four are valid modifications to synchronize access to the valid list between threads t1 and t2?

- A. Replace line 1 with: Synchronized (t2) (t1.start()); synchronized(t1) (t2.start());)korrekte Schreibweise: synchronized (t2) {t1.start();} synchronized(t1) { t2.start();}
- B. Replace Line 2 with: static CopyWriteArrayList<Integer> list = new CopyWriteArrayList<>(); korrekte Schreibweise: static CopyOnWriteArrayList<Integer> list = new CopyOnWriteArrayList<>();
- C. Replace line 3 with: synchronized public static void addItem () {korrekte Schreibweise: synchronized public static void addItem () {
- D. Replace line 4 with: synchronized (list) (list.add(1));korrekte Schreibweise: synchronized (list) { (list.add(1); }
- E. Replace line 5 with: Synchronized public void run () {korrekte Schreibweise: synchronized public void run () {
- F. replace line 6 with: Synchronized (this) {for (in i = 0, i<5000, i++) WorkPool.addItem(); }korrekte Schreibweise: synchronized (this) {for (int i = 0; i<500; i++) WorkPool.addItem(); }
- G. Replace line 6 with: synchronized (bar) {for (int i= 0; i<5000; i++) WorkPool.addItem(); }korrekte Schreibweise: synchronized (bar) {for (int i= 0; i<500; i++) WorkPool.addItem(); }

Answer: BCD

Explanation: Away to create synchronized code is with synchronized statements.

Unlike synchronized methods, synchronized statements must specify the object that provides the intrinsic lock:

For example:

```

public void addName(String name) { synchronized(this) {
    lastName = name; nameCount++;
}
}
nameList.add(name);
}

```

In this example, the addName method needs to synchronize changes to lastName and nameCount, but also needs to avoid synchronizing invocations of other objects' methods. Without synchronized statements, there would have to be a separate, unsynchronized method for the sole purpose of invoking nameList.add.

Reference: The Java Tutorial, Intrinsic Locks and Synchronization

NEW QUESTION 142

Assuming the port statements are correct, which two (three?) code fragments create a one- byte file?

- A. OutputStream fos = new FileOutputStream(new File("/tmp/data.bin")); OutputStream bos = new BufferedOutputStream(fos); DataOutputStream dos = new DataOutputStream(bos); dos.writeByte(0); dos.close();

B. OutputStream fos = new FileOutputStream ("/tmp/data.bin"); DataOutputStream dos = new DataOutputStream(fos); dos.writeByte(0);dos.close();
 C. OutputStream fos = new FileOutputStream (new File ("/tmp/data.bin")); DataOutputStream dos = new DataOutputStream(fos);dos.writeByte(0); dos.close();
 D. OutputStream fos = new FileOutputStream ("/tmp/data.bin"); fos.writeByte(0); fos.close();

Answer: ABC

Explanation: B:Create DataOutputStream from FileOutputStream public static void main(String[] args) throws Exception { FileOutputStream fos = new FileOutputS tream("C:/demo.txt"); DataOutputStream dos = new DataOutputStream(fos); Note:

The FileOutputStream class is a subclass of OutputStream. You can construct a FileOutputStream object by passing a string containing a path name or a File object.

You can also specify whether you want to append the output to an existing file. public FileOutputStream (String path)

public FileOutputStream (String path, boolean append) public FileOutputStream (File file)

public FileOutputStream (File file, boolean append)

With the first and third constructors, if a file by the specified name already exists, the file will be overwritten. Toappend to an existing file, pass true to the second or fourth constructor.

Note 2:public class DataOutputStreamextends FilterOutputStreamimplements DataOutput A data output stream lets an application write primitive Java data types to an output stream in a portable way.

An application can then use a data input stream to read the data back in. Reference:java.io Class DataOutputStream

NEW QUESTION 146

Given the fragment:

```
class MyClass extends Thread {
    public OtherThread ot;
    MyClass (String title){
        super(title);
    }
    public static void main(String[] args) {
        MyClass a = new MyClass ("Thread A") ;
        MyClass b = new MyClass ("Thread B") ;
        a.setThread (b) ;
        b.setThread (a) ;
        a.start();
        b.start();
    }
    public void run(){
        // use variable " ot " to do time-consuming stuff
    }
    public void setThread ( Thread x ) {
        ot = (OtherThread) x;
    }
}
```

If thread a and thread b are running, but not completing, which two could be occurring?

- A. livelock
- B. deadlock
- C. starvation
- D. loose coupling
- E. cohesion

Answer: AB

Explanation: A: A thread often acts in response to the action of another thread. If the other thread's action is also a responseto the action of another thread, then livelock may result. A thread often acts in response to the action ofanother thread. If the other thread's action is also a response to the action of another thread, then livelock mayresult.

B: Deadlock describes a situation where two or more threads are blocked forever, waiting for each other.

NEW QUESTION 151

Given:

```
interface Event {

String getCategory();
}
public class CueSports {
public String getCategory() {
return "Cue sports";
}
}
public class Snooker extends CueSports implements Event { // Line 9
public static void main(String[] args) {
Event obj1 = new Snooker(); // Line 11
CueSports obj2 = new Snooker(); // Line 12
System.out.print(obj1.GetCategory() + ", " + obj2.getCategory()); //Line 13
}
}
```

What is the result?

- A. Cue sports, Cue sports
- B. Compilation fails at line 9
- C. Compilation fails at line 11
- D. Compilation fails at line 12
- E. Compilation fails at line 13

Answer: B

Explanation: Class Snooker is public. Should be declared in a separate file. // Line 9 getCategory() >>> GetCategory() Line 13

NEW QUESTION 154

Give:

```
Class Employee {

public int checkEmail() {/* . . . */}

public void sendEmail (String email) {/* . . . */}

public Boolean validateEmail(){/* . . . */}

public void printLetter (String letter) {/* . . . */}

}
```

Which is correct?

- A. Employee takes advantage of composition.
- B. Employee "has-an" Email.
- C. Employee "is-a" LetterPrinter.
- D. Employee has low cohesion.

Answer: D

Explanation: The relationship between Employee and e-mail is poorly implemented here. There is low cohesion.

Note:

Low cohesion is associated with undesirable traits such as being difficult to maintain, difficult to test, difficult to reuse, and even difficult to understand.

Cohesion is decreased if:

The functionalities embedded in a class, accessed through its methods, have little in common. Methods carry out many varied activities, often using coarsely-grained or unrelated sets of data. Disadvantages of low cohesion (or "weak cohesion") are: Increased difficulty in understanding modules.

Increased difficulty in maintaining a system, because logical changes in the domain affect multiple modules, and because changes in one module require changes in related modules. Increased difficulty in reusing a module because most applications won't need the random set of operations provided by a module.

Reference: Cohesion (computer science)

NEW QUESTION 157

Given the code fragment:

```
String s = "Java 7, Java 6";
Pattern p = Pattern.compile("Java.+\\d");
Matcher m = p.matcher(s);
while (m.find()) {
    System.out.println(m.group());
}
```

What is the result?

- A. Java 7
- B. Java 6
- C. Java 7, Java 6
- D. Java 7 java 6
- E. Java

Answer: C

Explanation: regex: Java / one or more anything !!! / ends with a digit so it is the source string

NEW QUESTION 162

Given:

```
class Car implements TurboVehicle, Steerable {
    // Car methods > interface Convertible
    {
        // Convertible methods
    }
public class SportsCar extends Car implements Convertible {
    }
}
```

Which statement is true?

- A. SportsCar must implement methods from TurboVehicle and steerable
- B. SportsCar must override methods defined by car.
- C. SportsCar must implement methods define by convertible.
- D. Instances of car can invoke convertible methods.

Answer: C

Explanation: To declare a class that implements an interface, you include an implements clause in the classdeclaration.

By convention, the implements clause follows the extends clause, if there is one. Here are the some point that must be considered while implementing an interface (or interfaces) into a javaclass.

A class implementing an interface must either implement all the methods of that interface otherwise known asthe abstract class.

A class in java may extend at most one superclass because java does not allow multiple inheritance, by it mayimplement more than one interface. Multiple inheritance in java is achieved through the interfaces. When aclass implements more than one interface then implement statement requires a comma- separated list ofinterfaces to be implement by that class.

NEW QUESTION 166

Given the following code fragment:

```
public static void main(String[] args) {  
    Connection conn = null;  
    Deque<String> myDeque = new ArrayDeque<>();  
    myDeque.add("one");  
    myDeque.add("two");  
    myDeque.add("three");  
    System.out.println(myDeque.remove());  
}
```

What is the result?

- A. Three
- B. One
- C. Compilation fails
- D. The program runs, but prints no output

Answer: B

Explanation: add

boolean add(E e)

Inserts the specified element into the queue represented by this deque (in other words, at the tail of this deque) if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an `IllegalStateException` if no space is currently available. When using a capacity-restricted deque, it is generally preferable to use `offer`.

This method is equivalent to `addLast(E)`.

`remove`

Retrieves and removes the head of the queue represented by this deque (in other words, the first element of this deque). This method differs from `poll` only in that it throws an exception if this deque is empty.

This method is equivalent to `removeFirst()`. Returns:

The head of the queue represented by this deque

NEW QUESTION 169

Given the classes:

```
class Pupil {  
    String name = "unknown";  
    public String getName() { return name; }  
}
```

```
class John extends Pupil {  
    String name = "John";  
}
```

```
class Harry extends Pupil {  
    String name = "Harry";  
    public String getName() { return name; }  
}
```

```
public class Director {  
    public static void main(String[] args) {  
        Pupil p1 = new John();  
        Pupil p2 = new Harry();  
        System.out.print(p1.getName() + " ");  
        System.out.print(p2.getName());  
    }  
}
```

What is the result?

- A. John Harry
- B. unknown Harry
- C. john unknown
- D. unknown unknown
- E. Compilation fails.
- F. An exception is thrown at runtime.

Answer: B

Explanation: getName() is missing in John, hence Pupils getName() is invoked and the String in Pupils scope returned.

NEW QUESTION 171

.....

THANKS FOR TRYING THE DEMO OF OUR PRODUCT

Visit Our Site to Purchase the Full Set of Actual 1Z0-804 Exam Questions With Answers.

We Also Provide Practice Exam Software That Simulates Real Exam Environment And Has Many Self-Assessment Features. Order the 1Z0-804 Product From:

<https://www.2passeasy.com/dumps/1Z0-804/>

Money Back Guarantee

1Z0-804 Practice Exam Features:

- * 1Z0-804 Questions and Answers Updated Frequently
- * 1Z0-804 Practice Questions Verified by Expert Senior Certified Staff
- * 1Z0-804 Most Realistic Questions that Guarantee you a Pass on Your FirstTry
- * 1Z0-804 Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year