TESTPDF 認證大師

台灣、香港服務站 | 認證考試題庫

http://www.testpdf.net

為期一年的免費更新服務

Exam : **1z0-804**

Title : Java SE 7 Programmer II Exam

Vendor : Oracle

Version : DEMO

NO.1 Which is a factory method from the java.text.NumberFormat class?
A.format (long number)
B.getInstance()
C.getMaxiraumFractionDigits ()
D.getAvailableLocales ()
E.isGroupingUsed()
Answer: B
Explanation:
To obtain a NumberFormat for a specific locale, including the default locale, call one of NumberFormat's factory methods, such as getInstance().
Reference:java.textClass DecimalFormat

NO.2 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 John 4 Jack
D.1 Sam 3 John 4 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.

NO.3 Given:

```
class MarkOutOfBoundsException extends ArrayIndexOutOfBoundsException {}

public class Test {
   public void verify(int[] arr) throws ArrayIndexOutOfBoundsException {
      for (int i = 1; i <= 3; i++) {
         if(arr[i] > 100)
         throw new MarkOutOfBoundsException();
         System.out.println(arr[i]);
      }
   }
   public static void main(String[] args) {
      int[] arr = {105,78,56};
      try {
         new Test().verify(arr);
      } catch (ArrayIndexOutOfBoundsException | MarkOutOfBoundsException e) {
         System.out.print(e.getClass());
      }
   }
}
```

What is the result?
A.Compilation fails.
B.78 class java.lang.Array.IndexOutOfBoundException
C.class MarkOutOfBoundException
D.class java.lang.arrayIndexOutOfBoundException
Answer: A
Explanation:
The exception MarkOutOfBoundsException is already caught by the alternative ArrayIndexOutOfBoundsException

NO.4 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: B,F
Explanation:
B: The relationship modeled by composition is often referred to as the "has-a" relationship. Here Flommer has a String.
E: The has-a relationship has an encapsulation feature (like private or protected modifier used before each member 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 each member field or method).

NO.5 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.

NO.6 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

NO.7 Given the code fragment:

```
public class App {
   public static void main (String [] args){
      Path path = Paths.get("C:\\education\\institute\\student\\report.txt");
      System.out.printf("get.Name(0): %s \n", path.getName(0));
      System.out.printf("subpath(0, 2): %s", path.subpath (0, 2));
   }
}
```

What is the result?
A.getName (0): C:\
subpath (0, 2): C:\education\report.txt
B.getName(0): C:\ subpth(0, 2): C:\education
C.getName(0): education subpath (0, 2): education\institute
D.getName(0): education subpath(0, 2): education\institute\student
E.getName(0): report.txt subpath(0, 2): insritute\student
Answer: C
Explanation:
Example:
Path path = Paths.get("C:\\home\\joe\\foo");
getName(0)
-> home
subpath(0,2)
Reference: The Java Tutorial, Path Operations

NO.8 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: C,E
Explanation:
Thread t2 is executed. Execution of T2 starts executionen of t1. Greeting is printed during the execution of t1.

NO.9 Given the code fragment:

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.concurrent.CopyOnWriteArraySet;

public class Rank {
   static CopyOnWriteArraySet<String> arr = new CopyOnWriteArraySet<>();
   static void verify() {
      String var ="";
      Iterator<String> e=arr.iterator();
      while (e.hasNext()) {
         var = e.next();
         if(var.equals("A"))
            arr.remove(var);
      }
   }
   public static void main (String[] args) {
      ArrayList<String> list1 = new ArrayList<>();
      list1.add("A"); list1.add("B");
      ArrayList<String> list2 = new ArrayList<>();
      list1.add("A"); list1.add("D");
      arr.addAll(list1);
      arr.addAll(list2);
      verify();
      for(String var : arr)
         System.out.print(var + " ");
   }
}
```

What is the result?
A.Null B D
B.Null B null D
C.B D
D.D
E.An exception is thrown at runtime
Answer: C

NO.10 Which two properly implement a Singleton pattern?
A.class Singleton {
private static Singleton instance;
private Singleton () {}

```
public static synchronized Singleton getInstance() {
if (instance == null) {
instance = new Singleton ();
}
return instance;
}
}
B.class Singleton {
private static Singleton instance = new Singleton();
protected Singleton () {}
public static Singleton getInstance () {
return instance;
}
}
C.class Singleton {
Singleton () {}
private static class SingletonHolder {
private static final Singleton INSTANCE = new Singleton ();
}
public static Singleton getInstance () {
return SingletonHolder.INSTANCE; } }
D.enum Singleton {
INSTANCE;
}
```

Answer: A,D

Explanation:

A: Here the method for getting the reference to the SingleTon object is correct.

B: The constructor should be private

C: The constructor should be private

Note: Java has several design patterns Singleton Pattern being the most commonly used. Java Singleton pattern belongs to the family of design patterns, that govern the instantiation process. This design pattern proposes that at any time there can only be one instance of a singleton (object) created by the JVM.

The class's default constructor is made private, which prevents the direct instantiation of the object by others (Other Classes). A static modifier is applied to the instance method that returns the object as it then makes this method a class level method that can be accessed without creating an object.

OPTION A == SHOW THE LAZY initialization WITHOUT DOUBLE CHECKED LOCKING TECHNIQUE ,BUT ITS CORRECT OPTION D == Serialzation and thraead-safety guaranteed and with couple of line of code enum Singleton pattern is best way to create Singleton in Java 5 world.

AND THERE ARE 5 WAY TO CREATE SINGLETON CLASS IN JAVA 1>>LAZY LOADING (initialization) USING SYCHRONIZATION 2>>CLASS LOADING (initialization) USING private static final Singleton instance = new

Singleton(); 3>>USING ENUM 4>>USING STATIC NESTED CLASS

5>>USING STATIC BLOCK

AND MAKE CONSTRUCTOR PRIVATE IN ALL 5 WAY.

NO.11 Given the fragment: If thread a and thread b are running, but not completing, which two could be occurring?

```
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;
   }
}
```

A.livelock
B.deadlock
C.starvation
D.loose coupling
E.cohesion
Answer: A,B
Explanation:
A: 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. 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.
B: Deadlock describes a situation where two or more threads are blocked forever, waiting for each other.

NO.12 Given: javac Test.java java ea Test

```
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:
What is the result?
A.Compilation fails
B.Standard Edition Enterprise Edition Micro Edition
C.Standard Edition class 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 (class java.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 a specified 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 enables assertions.
Note 2:
An assertion is a statement in the JavaTM 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 interprete the Java bytecodes.
Reference:java - the Java application launcher
Reference:java.langClass AssertionError