**Vendor:** Oracle

**Exam Code:** 1Z0-808

**Exam Name:** Java SE 8 Programmer I

## Question 61—Question 70



**QUESTION 61**
Which statement best describes encapsulation?

A. Encapsulation ensures that classes can be designed so that only certain fields and methods of an object are accessible from other objects.
B. Encapsulation ensures that classes can be designed so that their methods are inheritable.
C. Encapsulation ensures that classes can be designed with some fields and methods declared as abstract.
D. Encapsulation ensures that classes can be designed so that if a method has an argument MyType x, any subclass of MyType can be passed to that method.

**Answer:** A
**Explanation:**
http://www.tutorialspoint.com/java/java_encapsulation.htm

**QUESTION 62**
Given the code fragment from three files:

SalesMan.java:

```
package sales;
public class SalesMan { }
```

Product.java:

```
package sales.products;
public class Product { }
```

Market.java:

```
1. package market;
2. // insert code here
3. public class USMarket {
4.     SalesMan sm;
5.     Product p;
6. }
```

Which code fragment, when inserted at line 2, enables the code to compile?

A)
```
import sales.*;
```

B)
```
import java.sales.products.*;
```

C)
```
import sales;
import sales.products;
```

D)
```
import sales.*;
import products.*;
```

E)
```
import sales.*;
import sales.products.*;
```

A. Option A
B. Option B
C. Option C
D. Option D
E. Option E

**Answer:** E

**Explanation:**

https://docs.oracle.com/javase/tutorial/java/package/usepkgs.html

**QUESTION 63**

Given the following class:

```
public class CheckingAccount {
    public int amount;
    public CheckingAccount(int amount){
        this.amount = amount;
    }
    public int getAmount(){
        return amount;
    }
    public void changeAmount(int x){
        amount += x;
    }
}
```

And given the following main method, located in another class:

```
public static void main(String[] args) {
    CheckingAccount acct = new CheckingAccount((int)(Math.random()*1000));
    //line n1
    System.out.println(acct.getAmount());
}
```

Which three lines, when inserted independently at line n1, cause the program to print a o balance?

A. this.amount = 0;
B. amount = 0;
C. acct (0) ;
D. acct.amount = 0;
E. acct. getAmount () = 0;
F. acct.changeAmount(0);
G. acct.changeAmount(-acct.amount);
H. acct.changeAmount(-acct.getAmount());

**Answer:** DGH

**Explanation:**

A and B don't compile because there isn't a variable amount in method main.
C is wrong because we can't call the constructor acct directly.
E is wrong because we can't make a method on acct equal to 0.
F is wrong because does not change variable amount of class CheckingAccount.

**QUESTION 64**
Given the code fragment:

```
String shirts[][] = new String[2][2];
shirts[0][0] = "red";
shirts[0][1] = "blue";
shirts[1][0] = "small";
shirts[1][1] = "medium";
```

Which code fragment prints red: blue: small: medium?

A)
```
for (int index = 1; index < 2; index++) {
    for (int idx = 1; idx < 2; idx++) {
        System.out.print(shirts[index][idx] + ":");
    }
}
```

B)
```
for (int index = 0; index < 2; ++index) {
    for (int idx = 0; idx < index; ++idx) {
        System.out.print(shirts[index][idx] + ":");
    }
}
```

C)
```
for (String c : colors) {
    for (String s : sizes) {
        System.out.println(s + ":");
    }
}
```

D)
```
for (int index = 0; index < 2;) {
    for (int idx = 0; idx < 2;) {
        System.out.print(shirts[index][idx] + ":");
        idx++;
    }
    index++;
}
```

A. Option A
B. Option B
C. Option C
D. Option D

**Answer:**D

**QUESTION 65**
Given the code fragment:

```
public class Test{

    void readCard(int cardNo) throws Exception {
        System.out.println("Reading Card");
    }

    void checkCard(int cardNo) throws RuntimeException { // line n1
        System.out.println("Checking Card");
    }

    public static void main(String[] args) {
        Test ex = new Test();
        int cardNo = 12344;
        ex.checkCard(cardNo);                    //line n2
        ex.readCard(cardNo);                     //line n3
    }
}
```

What is the result?

A. Reading Card
   Checking Card
B. Compilation fails only at line n1.
C. Compilation fails only at line n2.
D. Compilation fails only at line n3.
E. Compilation fails at both line n2 and line n3.

**Answer:** D
**Explanation:**
Exception is a checked exception so we are required to check it with try/catch or be declared in method main.

**QUESTION 66**
Given the code fragment:

```
public static void main(String[] args) {
    StringBuilder sb = new StringBuilder(5);
    String s = "";

    if (sb.equals(s)) {
        System.out.println("Match 1");
    } else if (sb.toString().equals(s.toString())) {
        System.out.println("Match 2");
    } else {
        System.out.println("No Match");
    }
}
```

What is the result?

A. Match 1
B. Match 2
C. No Match
D. A NullPointerException is thrown at runtime.

**Answer:** B
**Explanation:**
It will compare the string contents of the StringBuilder with string object.

**QUESTION 67**
Given:

```
package p1;
public class Acc {
    int p;
    private int q;
    protected int r;
    public int s;
}
```

Test.java:

```
package p2;
import p1.Acc;
public class Test extends Acc {
    public static void main(String[] args) {
        Acc obj = new Test();
    }
}
```

Which statement is true?

A. Both p and s are accessible by obj.
B. Only s is accessible by obj.
C. Both r and s are accessible by obj.
D. p, r, and s are accessible by obj.

**Answer:** B
**Explanation:**
Only s is accessible because it is the only public member of class Acc.

**QUESTION 68**
Given:

Base.java:

```
class Base {
    public void test(){
        System.out.println("Base ");
    }
}
```

DerivedA.java:

```
class DerivedA extends Base {
    public void test(){
        System.out.println("DerivedA ");
    }
}
```

DerivedB.java:

```
class DerivedB  extends DerivedA {
    public void test(){
        System.out.println("DerivedB ");
    }
    public static void main(String[] args) {
        Base b1 = new DerivedB();
        Base b2 = new DerivedA();
        Base b3 = new DerivedB();
        b1 = (Base) b3;
        Base b4 = (DerivedA) b3;
        b1.test();
        b4.test();
    }
}
```

What is the result?

A. Base
DerivedA

B. Base
DerivedB

C. DerivedB
DerivedB

D. DerivedB
DerivedA

E. A classcast Except ion is thrown at runtime.

**Answer:** C

**QUESTION 69**
Given the code fragment:

```
public static void main(String[] args) {
    ArrayList myList = new ArrayList();
    String[] myArray;
    try {
       while (true) {
           myList.add("My String");
       }
    }
    catch (RuntimeException re) {
       System.out.println("Caught a RuntimeException");
    }
    catch (Exception e) {
       System.out.println("Caught an Exception");
    }
    System.out.println("Ready to use");
}
```

What is the result?

A. Execution terminates in the first catch statement, and caught a RuntimeException is printed to the console.
B. Execution terminates In the second catch statement, and caught an Exception is printed to the console.
C. A runtime error is thrown in the thread "main".
D. Execution completes normally, and Ready to use is printed to the console.
E. The code fails to compile because a throws keyword is required.

**Answer:** C
**Explanation:**
while loop is an infinite loop so the program ends with an OutOfMemoryError.
This error can't be caught with Exception nor RuntimeException.
http://stackoverflow.com/questions/1692230/is-it-possible-to-catch-out-of-memory-exception-in-java

**QUESTION 70**

Given:

```
System.out.println("5 + 2 = " + 3 + 4);
System.out.println("5 + 2 = " + (3 + 4));
```

What is the result?

A) 5 + 2 = 34
   5 + 2 = 34

B) 5 + 2 + 3 + 4
   5 + 2 = 7

C) 7 = 7
   7 + 7

D) 5 + 2 = 34
   5 + 2 = 7

A. Option A
B. Option B
C. Option C
D. Option D

**Answer:** D

**Explanation:**

If neither operand of + is a reference to a String object, the operator is the arithmetic addition operator, not the string concatenation operator. Note that Java does not allow a program to define overloaded operators. However, the language defines the + operator to have a meaning that is fundamentally different from arithmetic addition if at least one of its operands is a String object.

The way in which Java decides if + means arithmetic addition or string concatenation means that the use of parentheses can alter the meaning of the + operator.

See “String Concatenation Operator +” at http://oponet.stsci.edu/web/documentation/Java%20Reference%20Library%201.02/langref/ch04_06.htm