Μεθοδολογία Προγραμματισμού

Εισαγωγή στη Java

1. (1 Μονάδες) Ο τελεστής `instanceof` χρησιμοποιείται στη Java για να ελέγξουμε τον τύπο μιας μεταβλητής. Τι πιστεύετε ότι θα τυπώσει ο παρακάτω κώδικας;

```
Object o = new Object();

if (o instanceof Object) {
    System.out.println("o␣is␣an␣Object");
}
if (o instanceof String) {
    System.out.println("o␣is␣also␣a␣String");
}

String s = "";
if (s instanceof Object) {
    System.out.println("s␣is␣an␣Object");
}
if (s instanceof String) {
    System.out.println("s␣is␣also␣a␣String");
}
```

Α΄. o is an Object
s is an Object

Β΄. o is an Object
s is also a String

Γ΄. o is an Object
o is also a String
s is an Object
s is also a String

Δ΄. o is an Object
s is an Object
s is also a String

Ε΄. Καμία απάντηση δεν είναι σωστή

2. (1 Μονάδες) Δείτε τον ακόλουθο κώδικα:

```
public class aClass {
    static int x;
    public static void setValue(final int value)
    {
       x = value;
    }

    public static void setOtherValue1(final int value)
    {
        value++;
    }

    public static void setOtherValue2(final String aString)
    {
        aString = new String();
    }
 }
```

   Α΄. Ο κώδικας σε όλες τις μεθόδους είναι σωστός

   Β΄. Ο κώδικας δεν είναι σωστός στη μέθοδο setOtherValue1() και setOtherValue2(). Δε θα γίνει καν compile.

   Γ΄. Ο κώδικας δεν είναι σωστός στη μέθοδο setOtherValue1() και setOtherValue2(). Θα γίνει compile αλλά δε θα τρέξει.

   Δ΄. Ο κώδικας δεν είναι σωστός στη μέθοδο setOtherValue1(). Θα γίνει compile αλλά δε θα τρέξει.

   Ε΄. Ο κώδικας δεν είναι σωστός στη μέθοδο setOtherValue1(). Δε θα γίνει καν compile.

3. (1 Μονάδες) Δείτε τον ακόλουθο κώδικα:

```
 //file ClassOne.java
package package1;

class ClassOne {
}
 //file ClassTwo.java
package package2;

class ClassTwo {
   ClassOne classOne;
}
```

   Α΄. Ο κώδικας δεν έχει κανένα πρόβλημα

   Β΄. Ο κώδικας θα ήταν σωστός αν η δεύτερη κλάση ξαναγραφόταν έτσι:

```
package package2;
import package1.ClassOne

class ClassTwo {
   ClassOne classOne;
}
```

Γ΄. Ο κώδικας δεν είναι σωστός. Ο κώδικας θα ήταν σωστός αν η δεύτερη κλάση ξαναγραφόταν έτσι:

```
package package2;
import package1.ClassOne

public class ClassTwo {
   ClassOne classOne;
}
```

Δ΄. Ο κώδικας δεν είναι σωστός. Αν προσπαθήσουμε να κάνουμε compile θα πάρουμε το μήνυμα:

Error:(4, 5) java: cannot find symbol
symbol: class ClassOne
location: package2.ClassTwo

Το πρόβλημα είναι ότι η κλάση ClassOne δεν είναι public και ανήκει σε διαφορετικό πακέτο.

Ε΄. Ο κώδικας δεν είναι σωστός. Αν προσπαθήσουμε να κάνουμε compile θα πάρουμε το μήνυμα:

Error:(4, 5) java: cannot find symbol
symbol: class ClassOne
location: package2.ClassTwo

Το πρόβλημα είναι ότι η κλάση ClassOne ανήκει σε διαφορετικό πακέτο.

4. (1 Μονάδες) Δύο κλάσεις που είναι στον ίδιο φάκελο (directory) μπορούν να ανήκουν σε διαφορετικά πακέτα.

   Α΄. Σωστό

   Β΄. Λάθος

5. (1 Μονάδες) Δείτε το ακόλουθο κώδικα

```
public class Main {
    public static void main(String[] args) {
        float a = 1.89f;
        System.out.println(11 * a);
    }
}
```

Τι πιστεύετε ότι θα τυπώσει;

Α΄. 20.8

Β΄. 20.789999 ή κάτι παρόμοιο. Εξαρτάται από το μηχάνημά μας

Γ΄. Θα βγάλει λάθος γιατί δε μπορούμε η System.out.println περιμένει ένα String ως όρισμα.

6. (1 Μονάδες) Δείτε τον ακόλουθο κώδικα. Τι θα τυπώσει;

```
public class ClassOne {
    private int x;
    private String s;
    void somethingSilly() {
        System.out.println(x);
        System.out.println(s);
    }
}
```

Α΄. Τυχαίες τιμές. Εξαρτάται από τι έτυχε να υπάρχει στη μνήμη μας εκείνη τη στιγμή.

Β΄. Θα δημιουργηθεί λάθος γιατί το πεδίο s έχει την τιμή null

Γ΄. Θα τυπώσει

0
null

Δ΄. Δε θα μπορέσει να γίνει compile, γιατί δεν έχουν δοθεί αρχικές τιμές στα πεδία. Θα εμφανιστεί το μήνυμα:

Error:(2, 4) java: variable x might not have been initialized
Error:(3, 4) java: variable s might not have been initialized

7. (1 Μονάδες) Δείτε τον ακόλουθο κώδικα. Τι θα τυπώσει;

```
public class ClassOne {
    void somethingSilly() {
        int x;
        String s;
        System.out.println(x);
        System.out.println(s);
    }
}
```

Α΄. Τυχαίες τιμές. Εξαρτάται από τι έτυχε να υπάρχει στη μνήμη μας εκείνη τη στιγμή.

Β΄. Θα δημιουργηθεί λάθος γιατί το πεδίο s έχει την τιμή null

Γ΄. Θα τυπώσει

0
null

Δ΄. Δε θα μπορέσει να γίνει compile, γιατί δεν έχουν δοθεί αρχικές τιμές στα πεδία. Θα εμφανιστεί το μήνυμα:

Error:(5, 28) java: variable x might not have been initialized
Error:(6, 28) java: variable s might not have been initialized

8. (1 Μονάδες) Έχετε αυτήν την κλάση

```
 package gr.teicm;
public class Main {

    public static void main(String[] args) {
        System.out.println(”Hello␣world’’);
}
```

και την κάνετε compile έτσι:

javac Main.java

ενώ στη συνέχεια προσπαθείτε να την τρέξετε έτσι:

java Main

Τι από τα παρακάτω είναι σωστό;

Α΄. Το πρόγραμμα θα τρέξει και θα τυπώσει: Hello World

Β΄. Θα εμφανιστεί το μήνυμα:

Error: Could not find or load main class Main

γιατί το όνομα της κλάσης μας δεν είναι Main αλλά gr.teicm.Main. Επομένως για να το τρέξουμε θα πρέπει να δώσουμε την εντολή,

java gr.teicm.Main

Γ΄. Θα εμφανιστεί το μήνυμα:

Error: Could not find or load main class Main

γιατί το όνομα της κλάσης μας δεν είναι Main αλλά gr.teicm.Main. Επομένως για να το τρέξουμε θα πρέπει να βάλουμε το αρχείο Main.class στο φάκελλο gr\teicm και στη συνέχεια από τον αρχικό φάκελο να δώσουμε την εντολή,

java gr.teicm.Main

9. (2 Μονάδες) Ο ακόλουθος κώδικας γράφει τιμές σε δύο πίνακες, έναν μονοδιάστατο και έναν δισδιάστατο. Πιο συγκεκριμένα, γράφουμε REPETITIONS φορές στον έναν πίνακα και στον άλλο και κρατάμε τον ελάχιστο χρόνο που απαιτήθηκε για το σύνολο των εγγραφών σε κάθε πίνακα. Πιστεύετε ότι υπάρχουν διαφορές στην ταχύτητα; Ποιος πίνακας νομίζετε ότι θα έχει πιο γρήγορη πρόσβαση; Αν γράφατε σε C πιστεύετε ότι θα είχατε το ίδιο αποτέλεσμα;

```
public class Main {
    final static int REPETITIONS = 100;
    final static int SIZE1 = 10000;
    final static int SIZE2 = 10000;
    static int[] dim1Array = new int[SIZE1*SIZE2];
    static int[][] dim2Array = new int[SIZE1][SIZE2];

    public static void main(String[] args) {
      long minDuration = Long.MAX_VALUE;

        for (int i = 0; i< REPETITIONS; i ++) {
            long duration = writeDim1Array();
            if (duration < minDuration) {
                minDuration = duration;
            }
        }
        System.out.println("Dim1Array␣write␣time:" + minDuration);

        minDuration = Long.MAX_VALUE;;
        for (int i = 0; i< REPETITIONS; i ++) {
            long duration = writeDim2Array();
            if (duration < minDuration) {
                minDuration = duration;
            }
        }
        System.out.println("Dim2Array␣write␣time:" + minDuration);
    }

    public static long writeDim1Array()
    {
        int r = 0;
        long start = System.nanoTime();
        for (int i = 0; i < dim1Array.length; i++) {
            dim1Array[i] =  r++;
        }
        long finish = System.nanoTime();
        return finish-start;
    }

    public static long writeDim2Array() {
        int r = 0;
        long start = System.nanoTime();
        for (int i = 0; i < dim2Array.length; i++) {
            for (int j = 0; j < dim2Array[i].length; j++) {
                dim2Array[i][j] = r++;
            }
        }
        long finish = System.nanoTime();
        return finish-start;
    }
}
```

10. (2 Μονάδες) Τρέξτε τον παραπάνω κώδικα έτσι:

    java -Djava.compiler=NONE Main

    Υπάρχει διαφορά; Που οφείλεται; *Βοήθεια: Κοιτάξτε τι είναι το JIT Just In Time Compilation*