Μεθοδολογία Προγραμματισμού

Εισαγωγή στη Java - Java interfaces/abstract classes

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

```
public interface AnInterface {
     int x = 0;

     void doThisLater();
     default void doThisNow() {
         System.out.println("In the interface x is " + x);
     }
}
public class AnImplementation implements AnInterface {
    int x = 1;
    @Override
    public void doThisLater() {
        System.out.println("In The class x is:" + x);
    }
}
public class Main {
    public static void main(String[] args) {
           AnInterface anInterface = new AnImplementation();

           anInterface.doThisLater();
           anInterface.doThisNow();
    }
}
```

Α′. `In The class x is 1`
`In the interface x is 1`

Β′. In The class x is 1
In the interface x is 0

Γ′. In The class x is 0
In the interface x is 0

Δ′. Θα βγάλει λάθος γιατί δεν επιτρέπεται ένα **interface** να έχει ιδιότητες

Ε′. Θα βγάλει λάθος γιατί δεν επιτρέπεται ένα **interface** να έχει υλοποιημένες μεθόδους

2. (1 Μονάδες) Έστω ότι αλλάζαμε τον κώδικα ως εξής:

```
public class AnImplementation implements AnInterface {
    @Override
    public void doThisLater() {
        x = 100;
        System.out.println("In␣The␣class␣x␣is␣" + x);
    }
}
```

Αν το τρέχαμε θα τύπωνε:

Α΄. In The class x is 100
In the interface x is 0

Β΄. In The class x is 0
In the interface x is 0

Γ΄. Ο κώδικας δε θα κάνει compile γιατί η μεταβλητή x, όπως χρησιμοποιείται στην κλάση, προέρχεται από το **interface** και είναι **final**

Δ΄. Ο κώδικας θα κάνει compile αλλά δε θα τρέξει βγάζοντας RuntimeError

Ε΄. Τίποτα από τα παραπάνω δεν είναι σωστό.

3. (1 Μονάδες) Έστω ότι αλλάζαμε τον κώδικα ως εξής:

```
public class AnImplementation implements AnInterface {
    @Override
    public void doThisLater() {
        System.out.println("In␣The␣class␣x␣is␣" + x);
    }
    @Override
    public void doThisNow() {
        System.out.println("doThisNow␣is␣defined␣in␣the␣class␣now");
    }
}
```

Α΄. Θα τύπωνε:

In The class x is 0
doThisNow is defined in the class now

Β΄. Θα έβγαζε λάθος γιατί δεν είναι δηλωμένο πουθενά το x

Γ΄. Θα έβγαζε λάθος γιατί δεν επαναορίζουμε τη μέθοδο doThisNow και αυτό δεν επιτρέπεται

Δ΄. Το (β) και το (γ)

Ε΄. Τίποτα από τα παραπάνω

4. (1 Μονάδες) Έστω ότι αλλάζαμε τον κώδικα του **interface** ως εξής:

```
public interface AnInterface {
     int x = 0;
     void doThisLater();
     default void doThisNow() {
         System.out.println("In␣the␣interface␣x␣is␣" + x);
     }
     static void doStatic() {
         System.out.println("In␣the␣static␣method␣x␣is␣" + x);
     }
}
```

Αν προσπαθούσαμε να ξαναορίσουμε τη μέθοδο doStatic() στην κλάση όπως κάναμε με τη doThisLater()

Α΄. Δε θα έχουμε κανένα πρόβλημα. Αρκεί να μη βάλουμε το annotation O@verride πιο πριν. Αλλά αυτό σημαίνει ότι δεν επαναορίζουμε τη μέθοδο, απλώς ότι δηλώνουμε μια νέα με το ίδιο όνομα.

Β΄. Η μέθοδος doStatic() δε μπορεί να αναφέρεται στη x γιατί ο compiler δεν ξέρει αν το x ανήκει στο **interface** ή στην κλάση. Άρα θα βγάλει λάθος στον ορισμό του **interface**.

Γ΄. Αν δεν υπήρχε η αναφορά στο x δε θα είχαμε πρόβλημα να ξαναορίσουμε τη μέθοδο.

Δ΄. Θα έχουμε compile time error γιατί δεν επιτρέπεται ο επανα-ορισμός **static** μεθόδων

Ε΄. Τίποτα από τα παραπάνω

5. (1 Μονάδες) Έστω ότι η κλάση AnImplementation οριζόταν τώρα ως εξής:

```
public class AnImplementation implements AnInterface {
    @Override
    public void doThisLater() {
        System.out.println("This is the class");
    }

    static void doSomethingElse() {
        //Call doStatic here:

    }
}
```

Για να καλέσουμε τη doStatic() πως θα το κάναμε;

Α΄. Προσθέτωντας μια κλήση ως εξής: doStatic();

Β΄. Κάτι τέτοιο δεν επιτρέπεται. Δε μπορούμε να καλέσουμε τη doStatic()

Γ΄. AnInterface.doStatic(); γιατί οι **static** μέθοδοι είναι του **interface**

Δ΄. Ε'ιτε το (α) είτε το (γ)

Ε΄. Τίποτα από τα παραπάνω

6. (1 Μονάδες) Έστω ότι προσθέτουμε ένα νέο **interface** και αλλάζαμε την κλάση AnImplementation ως εξής:

```
public interface AnInterface2 {
    int x = 0;
    void doThisLater();
    default void doThisNow() {
        System.out.println("In the interface x is " + x);
    }
}
public class AnImplementation implements AnInterface, AnInterface2 {
    @Override
    public void doThisLater() {
        System.out.println("This is the class");
    }
}
```

Τι πιστεύετε ότι θα γίνει

Α΄. Δε θα έχουμε κανένα πρόβλημα

Β΄. Ο compiler δεν ξέρει ποιά από τις **default** υλοποιήσεις θα χρησιμοποιήσει. Επομένως θα έχουμε compile time error και για να το λύσουμε θα πρέπει να επαναορίσουμε την doThisNow() στην AnImplementation

Γ΄. Απαγορεύεται η υλοποίηση πολλών interfaces στη Java

Δ΄. Η υλοποίηση πολλών interfaces επιτρέπεται αλλά όχι όταν έχουν τα ίδια ονόματα μεθόδων.

7. (1 Μονάδες) Με βάση τον προηγούμενο κώδικα ποιές γραμμές πιστεύετε ότι είναι λάθος στον παρακάτω κώδικα;

```
1     public static void main(String[] args) {
2         AnImplementation anImplementation = new AnImplementation();
3
4         AnInterface anInterface = new AnInterface();
5         AnInterface2 interace22 = new AnInterface();
6
7         AnInterface2 interface2 = new AnImplementation();
8         AnInterface interface3 = new AnImplementation();
9     }
```

Α΄. Οι γραμμές 4-5 και 7-8

Β΄. Οι γραμμές 7-8

Γ΄. Οι γραμμές 4-5

Δ΄. Οι γραμμές 2, 7-8

Ε΄. Η γραμμή 5

8. (1 Μονάδες) Σε μία κλάση μπορεί να έχετε initializers (**static** ή όχι), κώδικα δηλαδή που εκτελείται μία φορά για την κλάση (αν είναι **static**) ή για το instance. Δείτε το ακόλουθο παράδειγμα.

```
class A
{
    {
        System.out.println(1);
    }
}
class B extends A
{
    {
        System.out.println(2);
    }
}

class C extends B
{
    {
        System.out.println(3);
    }
}
public class MainClass
{
    public static void main(String[] args)
    {
        C c = new C();
    }
}
```

Τι θα τυπώσει;

Α΄. 3
2
1

Β΄. 1
3

Γ΄. 3

Δ΄. 1
2
3

9. (1 Μονάδες) Τι θα τυπώσει το παρακάτω πρόγραμμα;

```
abstract class A
{
    abstract void firstMethod();
    void secondMethod()
    {
        System.out.println("SECOND");
        firstMethod();
    }
}
abstract class B extends A
{
    @Override
    void firstMethod()
    {
        System.out.println("FIRST");
        thirdMethod();
    }
    abstract void thirdMethod();
}
class C extends B
{
    @Override
    void thirdMethod()
    {
        System.out.println("THIRD");
    }
}
public class MainClass
{
    public static void main(String[] args)
    {
        C c = new C();
        c.firstMethod();
        c.secondMethod();
        c.thirdMethod();
    }
}
```

Α′. FIRST
THIRD
SECOND
FIRST
THIRD
THIRD

Β′. Δε θα κάνει compile

Γ′. FIRST
SECOND
THIRD
FIRST
THIRD
THIRD

Δ΄. FIRST
THIRD
SECOND
THIRD
THIRD

Ε΄. FIRST
SECOND
THIRD
SECOND
FIRST
THIRD
THIRD

10. (1 Μονάδες) Τι είναι η αρχή υποκατάστασης της Liskov[1]; Αν είχαμε μια κλάση Rectangle και μια κλάση Square που κληρονομούσε από τη Rectangle θα είχαμε πρόβλημα με την αρχή υποκατάστασης της Liskov; Υπό ποιες προϋποθέσεις; (Αρχή αντικαστάσης Liskov. Αν το S είναι υποκλάση του T τότε θα πρέπει να μπορούμε να υποκαταστήσουμε οπουδήποτε χρησιμοποιούμε το με το S)

11. (Για το σπίτι) Τι είναι η αρχή Open-Closed (Open For Extension, Closed for Modifiation) του Meyer[2]; Βρείτε ένα παράδειγμα που παραβιάζει την αρχή.

[1] Μπάρμπαρα Λίσκοφ: Αμερικανίδα επιστήμονας πληροφορικής, νικήτρια του βραβείου Turing

[2] Bertrand Meyer Γάλλος επιστήμονας πληροφορικής, δημιουργός της γλώσσας Eiffel