ΕΡΓΑΣΤΗΡΙΟ 16

Interfaces and Abstract Classes

1. Τι εμφανίζει στην οθόνη η εκτέλεση της κλάσης Test?

```
public class Test {
	public static void main(String[] args) {
		new Circle9();
	}
}

abstract class GeometricObject {
	protected GeometricObject() {
		System.out.print("A");
	}

	protected GeometricObject(String color, boolean filled) {
		System.out.print("B");
	}
}

class Circle9 extends GeometricObject {
	/** Default constructor */
	public Circle9() {
		this(1.0);
		System.out.print("C");
	}

	/** Construct circle with a specified radius */
	public Circle9(double radius) {
		this(radius, "white", false);
		System.out.print("D");
	}

	/** Construct a circle with specified radius, filled, and color */
	public Circle9(double radius, String color, boolean filled) {
		super(color, filled);
		System.out.print("E");
	}
}
```

2. Ποια θα είναι η έξοδος του ακόλουθου προγράμματος;

```
interface One {
	String s = "FINAL";

	String methodONE();
}

interface Two {
	String methodONE();
}
```

```
abstract class Three {
	String s = "NOT FINAL";

	public abstract String methodONE();
}

class Four extends Three implements One, Two {
	public String methodONE() {
		String s = super.s + One.s;
		return s;
	}
}

public class MainClass {
	public static void main(String[] args) {
		Four four = new Four();
		System.out.println(four.methodONE());
		One one = four;
		System.out.println(one.s);
	}
}
```

3. Ποια θα είναι η έξοδος του ακόλουθου προγράμματος;

```
interface P {
	String p = "PPPP";
	String methodP();
}

interface Q extends P {
	String q = "QQQQ";
	String methodQ();
}

class R implements P, Q {
	public String methodP() {
		return q + p;
	}

	public String methodQ() {
		return p + q;
	}
}

public class MainClass1 {
	public static void main(String[] args) {
		R r = new R();

		System.out.println(r.methodP());

		System.out.println(r.methodQ());
	}
}
```

4. Ποια θα είναι η έξοδος του ακόλουθου προγράμματος;

```
interface A {
	String A = "AAA";

	String methodA();
}

interface B {
	String B = "BBB";

	String methodB();
}

class C implements A, B {
	public String methodA() {
		return A + B;
	}

	public String methodB() {
		return B + A;
	}
}

class D extends C implements A, B {
	String D = "DDD";

	public String methodA() {
		return D + methodB();
	}
}

public class MainClass2 {
	public static void main(String[] args) {
		C c = new C();

		System.out.println(c.methodA());

		System.out.println(c.methodB());

		c = new D();

		System.out.println(c.methodA());

		System.out.println(c.methodB());
	}

}
```

5. Στο προηγούμενο εργαστήριο σχεδιάσαμε μια κλάση, ShapeBasics, χωρίς την απαίτηση να δημιουργήσουμε αντικείμενα αυτής της κλάσης. Την σχεδιάσαμε σαν βασική κλάση από την οποία κληρονομούσαν άλλες κλάσεις όπως η Rectangle και Triangle. Όμως, ο

τρόπος σχεδίασης μας επιτρέπει να δηλώσουμε αντικείμενα αυτής της κλάσης όπως παρακάτω: ShapeBasics shapeVariable = new ShapeBasics()

- Μετατρέψτε την κλάση ShapeBasics σε μια αφαιρετική κλάση, και μετονομάστε την σε ShapeBase.
- Σχεδιάστε δυο νέες κλάσεις, RightArrow και LeftArrow οι οποίες κληρονομούν από τη ShapeBase. Αυτές οι κλάσεις θα είναι σαν τις κλάσεις Rectangle και Triangle, αλλά θα δημιουργήσουν αντικείμενα βέλη που δείχνουν δεξιά και αριστερά αντίστοιχα. Για παράδειγμα, το παρακάτω βέλος δείχνει προς δεξιά.

Το μέγεθος του βέλους καθορίζεται από δυο αριθμούς, ένα για το μήκος της «ουράς» και ένα για το πλάτος του τριγώνου, π.χ. το μήκος και πλάτος του παραπάνω βέλους είναι 12 και 7 αντίστοιχα. Το πλάτος του βέλους δεν μπορεί να είναι ζυγός αριθμός, έτσι οι κατασκευαστές και οι μέθοδοι settters πρέπει να ελέγχουν ότι είναι πάντα μονός. Γράψτε ένα πρόγραμμα δοκιμών για κάθε κλάση που ελέγχει όλες τις μεθόδους. Μπορείτε να υποθέσετε ότι το πλάτος του βέλους είναι τουλάχιστον 3.

6. (The Colorable interface) Σχεδιάστε μια διαπροσωπεία ονόματι Colorable η οποία έχει μια μέθοδο void howToColor(). Κάθε κλάση ενός «χρωματιστού» αντικείμενου (colorable object) πρέπει να υλοποιεί τη διαπροσωπεία Colorable. Σχεδιάστε μια κλάση Square που κληρονομεί από την κλάση GeometricObject και υλοποιεί τη Colorable. Υλοποιήστε τη μέθοδο howToColor() έτσι ώστε να εκτυπώνει το μήνυμα “Color all four sides”.

   Σχεδιάστε ένα διάγραμμα κλάσεων που περιλαμβάνει τις Colorable, Square, και GeometricObject. Γράψτε ένα πρόγραμμα δοκιμής το οποίο δημιουργεί ένα πίνακα με 5 GeometricObjects. Για κάθε αντικείμενο του πίνακα, το πρόγραμμα να εμφανίζει το εμβαδόν και να καλεστεί η μέθοδος howToColor() εάν είναι «χρωματιστό» αντικείμενο.