ΕΡΓΑΣΤΗΡΙΟ 12

Polymorphic behavior via method overriding

1. (Introduction to Polymorphism) Ας υποθέσουμε ότι έχετε 3 κλάσεις: A, B, και C, όπου η κλάση B κληρονομεί την κλάση A και η κλάση C κληρονομεί την κλάση B. Και οι τρεις κλάσεις υλοποιούν τη μέθοδο `void doIt()`. Η μέθοδος main() έχει το παρακάτω κώδικα:

```
public static void main(String[] args) {
     A x = new B();
     x.doIt();
}
```

   Ποια από τις μεθόδους `doIt`() εκτελείται και γιατί;

2. Ποια θα είναι η έξοδος του ακόλουθου προγράμματος;

```
class X {
    void method(int a) {
        System.out.println("ONE");
    }

    void method(double d) {
        System.out.println("TWO");
    }
}

class Y extends X {
    @Override
    void method(double d) {
        System.out.println("THREE");
    }
}

public class MainClass {
    public static void main(String[] args) {
        new Y().method(100);
    }
}
```

3. Σχεδιάστε την ιεραρχία κλάσεων: “Colored Rectangle inherits from Rectangle” λαμβάνοντας υπόψη τους ακόλουθους περιορισμούς: Η κλάση Rectangle έχει σαν χαρακτηριστικά το πλάτος και το ύψος. Η κλάση ColoredRectangle κληρονομεί από την κλάση Rectangle και έχει σαν χαρακτηριστικό το χρώμα. Ο κώδικας να δοκιμαστεί με την κλάση `TestRectangles`:

```
public class TestRectangles {
    public static void main(String[] args) {
      System.out.println("Test 1 :");
      Rectangle rect = new Rectangle(12.5, 4.0);
      System.out.println(rect);
      System.out.println();

      System.out.println("Test 2: ");
      // the type of rect1 is ColoredRectangle
      // the object contained in rect1 is of type ColoredRectangle
      ColoredRectangle rect1 = new ColoredRectangle(12.5, 4.0, "red");
      System.out.println(rect1);
      System.out.println();

      System.out.println("Test 3 :");
      // the type of rect2 is Rectangle
      // the object contained in rect2 is of type ColoredRectangle
      Rectangle rect2 = new ColoredRectangle(25.0/2, 8.0 / 2, new String("red"));
      System.out.println(rect2);

      System.out.println(rect1.equals(rect2)); // 1.
      System.out.println(rect2.equals(rect1)); // 2.
      System.out.println(rect1.equals(null)); // 3.
      System.out.println(rect.equals(rect1)); // 4.
      System.out.println(rect1.equals(rect)); // 5.
    }
}
```

Η έξοδος του προγράμματος πρέπει να είναι :

```
Test 1 :
Rectangle :
 width = 12.5
 height = 4.0

Test 2:
Rectangle :
 width = 12.5
 height = 4.0
 color = red

Test 3 :
Rectangle :
 width = 12.5
 height = 4.0
 color = red
true
true
false
false
false
```

Οι μέθοδοι toString() και equals() δεν πρέπει να έχουν επικάλυψη του κώδικα.

4. Θέλουμε να σχεδιάσουμε ένα σύνολο κλάσεων για να "μοντελοποιήσουμε" την οικογένεια των πτηνών (πουλιών). Για κάθε είδος πουλί θέλουμε να ορίσουμε μια μέθοδο, describe. Επίσης, θέλουμε να βάλουμε ένα μείγμα αντικειμένων "πουλιών" στο ίδιο πίνακα, και στη συνέχεια με ένα βρόγχο να κάνουμε προσπέλαση του πίνακα εφαρμόζοντας διαδοχικά τη μέθοδο describe για κάθε στοιχείο του πίνακα. Μπορείτε να χρησιμοποιήσετε τις παρακάτω κλάσεις (θα χρειαστεί να συμπληρώσετε κώδικα).

```
public class Blackbird { //κοτσύφι
    public void describe(){
  System.out.println(this + "I'm a blackbird");
    }
}

public class Pie { //καρακάξα
    public void describe(){
  System.out.println(this + " I'm a pie");
    }
}
```

   a. Στο πρόγραμμα σας, η μέθοδος main() θα δημιουργήσει ένα πίνακα με διαφορετικά πουλιά (π.χ. με κοτσύφια και καρακάξες) και εκτυπώνοντας τον πίνακα πρέπει να έχουμε σαν έξοδο:

```
Family of birds: I'm a blackbird
Family of birds: I'm a pie
Family of birds: I'm a pie
Family of birds: I'm a blackbird
Family of birds: I'm a pie
```

   b. Τι πρέπει να κάνετε για να έχετε την ακόλουθη έξοδο;

```
Family of birds: I'm a blackbird
Family of birds: I'm a pie
Family of birds: I'm a different bird
Family of birds: I'm a pie
```