Lab Assignment 5: Working with Inheritance, Interfaces and Packages

Course: Advanced Programming (EE423) Institution: IGEE, Boumerdes University

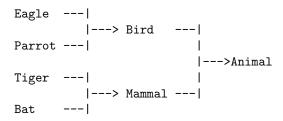
November 19, 2022

1 Objectives

Within this lab, you will, in the first part, learn about the organization of classes by putting them into packages. Classes should be grouped and put together inside packages in a coherent manner and each package should be given a meaningful and representative name of the group. You will also understand inheritance by coding it. In the second part, using inheritance, you will see two cases of polymorphisme namely overloading and overriding methods.

2 Assignment

2.1 Part 1: create the following structure of classes



- Put the class Bird and its sub-classes (Parrot and Eagle) in a package called animal.bird
- Put the class Mammal and its sub-classes (Bat and Tiger) in a package called animal.mammal
- Put the class Animal in a package called animal
- The packages bird and mammal are sub-packages of the package animal

Here is the code of the class Animal:

```
public class Animal {
  public Animal() { System.out.println("An animal is created"); }
  public void eat() {System.out.println("The animal is eating.");};
  public void run() {System.out.println("The animal is running.");};
  public void sleep() {System.out.println("The animal is sleeping.");};
}
```

Create a class called Mammal that extends the class Animal. Override all its functions. This time the messages should say: "The mammal is..." instead of "The animal is ...". Add a constructor to this class:

```
public Mammal() { System.out.println("A mammal is created"); }
```

Create a class called Bird that extends the class Animal. Override all its functions. This time the messages should say: "The bird is..." instead of "The animal is ...". Add a constructor to this class:

```
public Bird() { System.out.println("A bird is created"); }
```

Create a class called Bat that extends the class Mammal. Override all its functions. This time the messages should say: "The bat is..." instead of "The mammal is ...". Add a constructor to this class:

```
public Bat() { System.out.println("A bat is created"); }
```

Create a class called Tiger that extends the class Mammal. Override all its functions. This time the messages should say: "The tiger is..." instead of "The mammal is ...". Add a constructor to this class:

```
public Tiger() { System.out.println("A tiger is created"); }
```

Create a class called Parrot that extends the class Bird. Override all its functions. This time the messages should say: "The parrot is..." instead of "The bird is ...". Add a constructor to this class:

```
public Parrot() { System.out.println("A parrot is created"); }
```

Create a class called Eagle that extends the class Bird. Override all its functions. This time the messages should say: "The eagle is..." instead of "The bird is ...". Add a constructor to this class:

```
public Eagle() { System.out.println("An eagle is created"); }
```

2.1.1 Question 1

What is the output when we run the following code:

```
Mammal mammal = new Mammal();
```

Explain why.

2.1.2 Question 2

```
What is the output when we run the following code:
```

```
Parrot parrot = new Parrot();
```

Explain why.

2.1.3 Question 3

What is the output when we run the following code:

```
Tiger tiger= new Tiger();
```

Explain why.

2.1.4 Question 4

What is the output when we run the following code:

```
Animal animal = new Tiger();
animal.sleep();
```

Explain why.

Overload the function eat of Tiger by allowing a tiger to eat a parrot.

```
public void eat(Parrot parrot){
parrot.sleep();
System.out.println("A bad tiger just ate a parrot");
}
```

2.1.5 Question 5

What is the output when we run the following code:

```
Tiger tiger = new Tiger();
Parrot parrot = new Parrot();
tiger.eat(parrot);
```

Explain why.

Overload the function run of Parrot by allowing the parrot to run away when he sees a tiger.

```
public void run(Tiger tiger){
System.out.println("A tiger! Run run run!");
}
```

2.1.6 Question 6

What is the output when we run the following code:

```
Tiger tiger = new Tiger();
Parrot parrot = new Parrot();
parrot.run(tiger);
Explain why.
```

2.2 Part 2: Overloading and overriding methods

Create a class called Point

```
public class Point{
  private int x;
  private int y;
  public Point(int x, int y){ this.x = x; this.y = y;}
  public void setX(int x) {this.x = x;}
  public void setY(int y) {this.y = y;}
  public int getX(){return this.x;}
  public int getY(){return this.y;}
}
```

2.2.1 Question 1: Extending the class Point

Create a class called ColoredPoint which extends the class Point. Add an instance variable called color to the class ColoredPoint. Make color private and add its getter and setter methods.

2.2.2 Question 2: Calling a constructor of the super class

Add a constructor to the class ColoredPoint that takes as parameters three variables int x, int y, and a String color. This constructor should call the constructor of Point by passing the two variables x, y and after that it sets the value of color.

2.2.3 Question 3: Calling a constructor of the same class

Add a constructor to the class ColoredPoint that does not take any parameters. This constructor should call the constructor of ColoredPoint you created previously and should create an object with the values (0,0,"black")

2.2.4 Question 4: Method overloading

Add method called print(); to the class Point. This method should print the point in this format "Point(x,y)". Add another method called print(boolean pretty) but this time this function takes a parameter as input. If pretty is true, just print the point using the same format we saw previously "Point(x,y)" If pretty is false, just print "x,y"

2.2.5 Question 5: Method overriding

Add a method print to the class ColoredPoint that overrides the method print in the class Point. This time the print should have the format "Point(x,y,color)" Add another method to the class ColoredPoint that override the method print(boolean pretty) of the class Point. If pretty is true, just print the point using the format "Point(x,y,color)" If pretty is false, just print "x,y,color"

2.2.6 Question6: Calling methods

What is the output of this code:

```
Point pt = new Point(102,210);
pt.print(); pt.print(true); pt.print(false);

ColoredPoint cpt = new ColoredPoint(103,330,"black");
cpt.print(); cpt.print(true); cpt.print(false);

ColoredPoint cpt2 = new ColoredPoint();
cpt2.print(); cpt2.print(true); cpt2.print(false);
```

3 Conclusion

Packages are mandatory when working on big projects with plethora of classes. You can use them to group class coherently so that you can search and access classes easily. Inheritance is such a powerful concept that you must use to refactor code and structure it in a manner so that it becomes easy to maintain and upgrade.