Tutorial Week 03 – OO Revision Part 3

OBJECT ORIENTED CONCEPT # 3 - INHERITANCE

From www.javaworld.com/javaworld/jw-07-2001/jw-0706-java101.html

What is inheritance?

We encounter *inheritance* -- the ability to derive something specific from something generic -- in everyday life. For example, a blue Mazda RX-7 is a specific instance of the generic *car* category. Continuing with the example, a shiny red Dodge Ram Pickup traveling down the highway is a specific instance of the generic *truck* category. If you take the car and truck categories to another level, both categories relate to each other by being specific instances of the even more generic *vehicle* category. In other words, cars and trucks are vehicles.

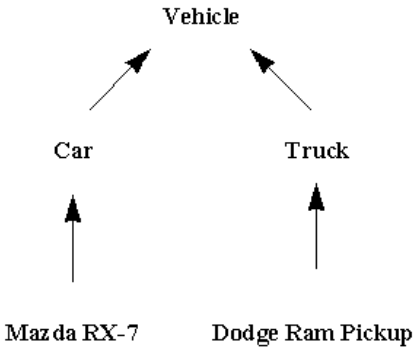


Figure 1. The relationships between a Mazda RX-7, a Dodge Ram Pickup, and the vehicle super category. The arrows point from specific entities to generic categories.

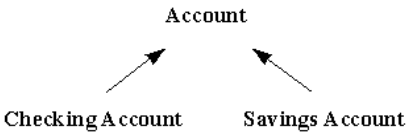
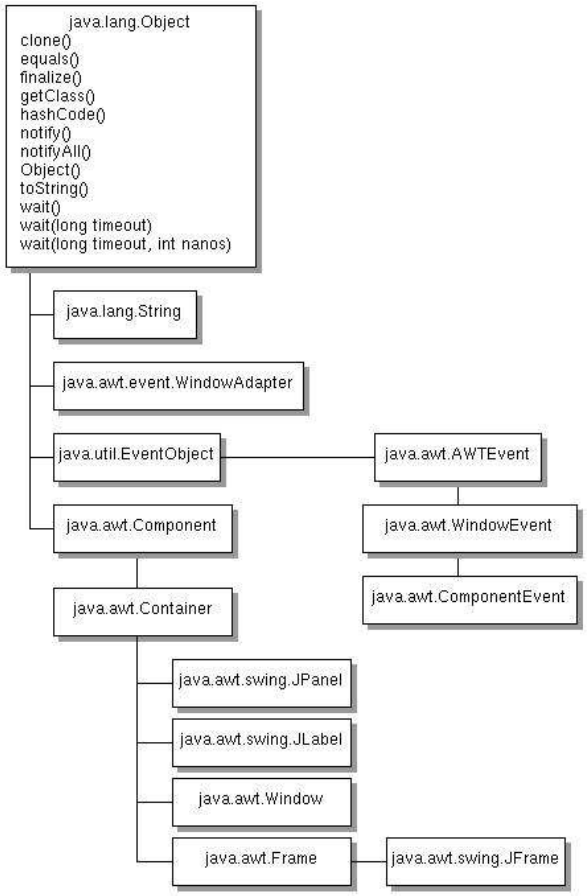


Figure 2. The relationships between a checking account, a savings account, and the bank account category. As in Figure 1, arrows point from the more specific to the more generic.

Consider a second example: You deposit some money into a savings account. Later, you write a check from your checking account to pay your Visa bill. At some point, money transfers from your checking account to Visa. What do your savings and checking accounts have in common? Nothing, apart from being bank accounts. In other words, savings and checking accounts inherit *state* (such as a current amount of money) and *behaviors* (such as "open the account" and "close the account") from the bank account category.

http://java.sun.com/developer/onlineTraining/Programming/BasicJava2/oo.html

Java Inheritance



One object-oriented concept that helps objects work together is inheritance. Inheritance defines relationships among classes in an object-oriented language. In the Java programming language, all classes descend from java.lang.Object and implement its methods

This diagram shows the class hierarchy as it descends from java.lang.Object for the classes in the user interface example above. The java.lang.Object methods are also shown because they are inherited and implemented by all of its subclasses, which is every class in the Java API libraries. java.lang.Object defines the core set of behaviors that all classes have in common.

As you move down the hierarchy, each class adds its own set of class-specific fields and methods to what it inherits from its superclass or superclasses. The java.awt.swing.JFrame class inherits fields and methods from java.awt.Frame, which inherits fields and methods from java.awt.Container, which inherits fields and methods from java.awt.Component, which finally inherits from java.lang.Object, and each subclass adds its own fields and methods as needed.

EXAMPLE

EmployeeThree.java (Superclass definition)

```
public class EmployeeThree {

   private String name;
   private double salary;
   private boolean status;

   EmployeeThree (String n, double s) {
      name = n;
      salary = s;
      status = true;
   }

   public double getSalary() {
      return salary;
   }

   public void setSalary(double s) {
      if (s < 25000)
         salary = s;
   }

   public String getName() {
      return name;
   }

   public String getStatus() {
      return status? "active":"resigned";
   }

   public void resigned() {
      status = false;
   }

   public void displayDetail() {
      System.out.println("=========================");
      System.out.println("NAME   : " + name);
      System.out.println("SALARY : " + salary);
      System.out.println("STATUS : " + getStatus());
   }
}
```

Officer.java (Subclass definition)

```
public class Officer extends EmployeeThree {

   private double petrol;

   public Officer(String n, double s, double p) {
      super(n,s);
      petrol = p;
```

```
   }
   public double getPetrol() {
      return petrol;
   }

   public void displayDetail() {
      super.displayDetail();
      System.out.println("PETROL : " + petrol);
   }
}
```

Manager.java (Subclass definition)

```
public class Manager extends EmployeeThree {

   private String car;

   public Manager(String n, double s, String c) {
      super(n,s);
      car = c;
   }

   public String getCar() {
      return car;
   }

   public void displayDetail() {
      super.displayDetail();
      System.out.println("CAR    : " + car);
   }
}
```

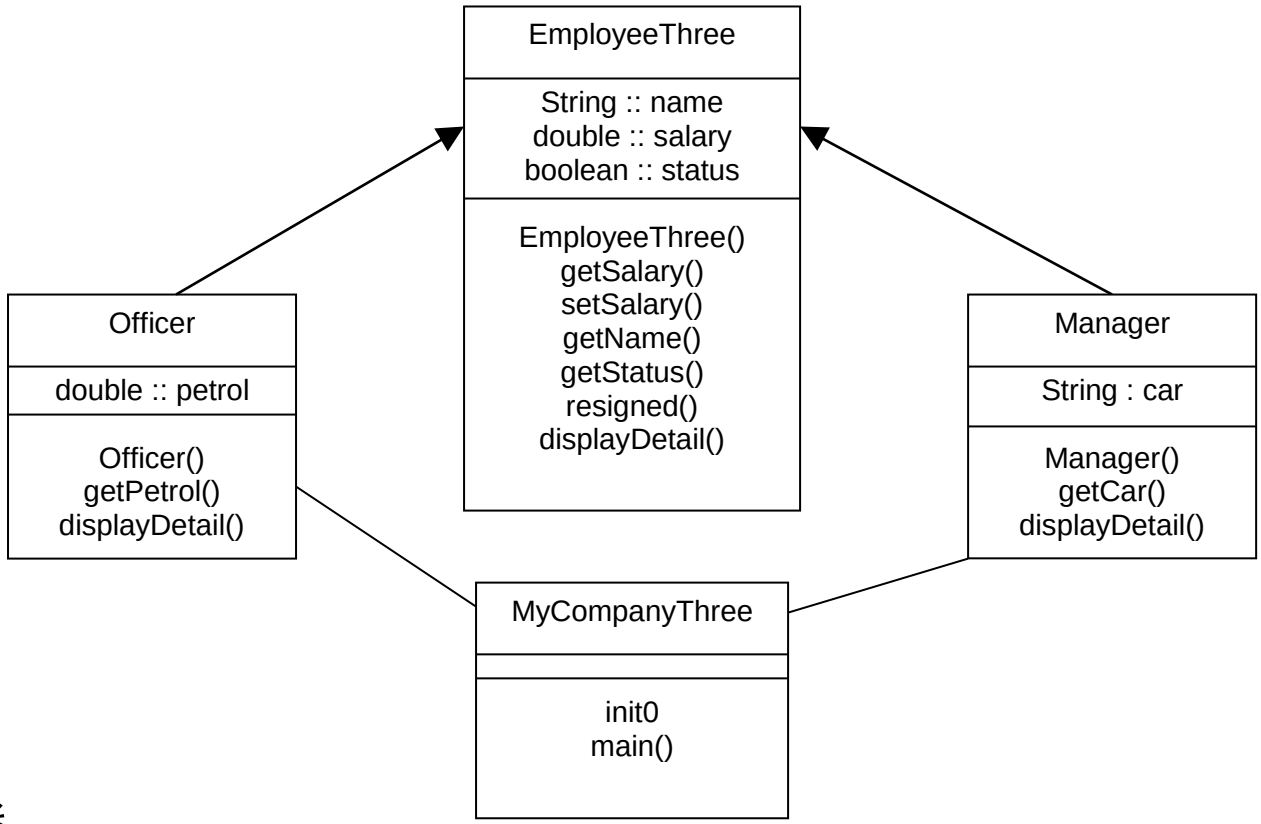
MyCompanyThree.java (Main program)

```
public class MyCompanyThree {

   public void init() {

      Officer staff1 = new Officer("Joyah", 3000, 300);
      Officer staff2 = new Officer("Mat", 2000, 100);
      Manager staff3 = new Manager("Minah", 8000,"Honda Accord");
      staff1.displayDetail();
      staff2.displayDetail();
      staff3.displayDetail();
      staff2.resigned();
      staff2.displayDetail();
   }

   public static void main(String args[]) {

      MyCompanyThree mct = new MyCompanyThree();
      mct.init();
   }
}
```

Result

```
=========================
NAME   : Joyah
SALARY : 3000.0
STATUS : active
PETROL : 300.0
=========================
NAME   : Mat
SALARY : 2000.0
STATUS : active
PETROL : 100.0
=========================
NAME   : Minah
SALARY : 8000.0
STATUS : active
CAR    : Honda Accord
=========================
NAME   : Mat
SALARY : 2000.0
STATUS : resigned
PETROL : 100.0
```

CLASS DIAGRAM



EXERCISE *Modify **EmployeeThree** program with the following requirements*

1. Add attribute YearBorn to Manager and Officer.
The year born data are as follow a) Joyah = 1965
b) Mat = 1974
c) Minah = 1983
2. Display details showing age assuming current year is 2014.
*3. Add class **Clerk.** Clerks are not entitled to company car or petrol allowance but they may get Cost of Living allowance (COLA).*
4. Add a clerk and show the details. Name "Selamat", Salary RM800, Year Born 1978, COLA RM50.
5. Redraw the class diagram to reflect the modified program.