COMP1102/8702 – Practical Class 9

Inheritance

Aims and Objectives

The aim of the practical is to provide a basis for experimentation with basic forms of inheritance

On successful completion of the practical you will have the ability to construct Java classes which extend/modify the behaviour of existing classes by overriding (redefining) existing methods or including new methods.

Task 1

Start *IntelliJ* and open the project named "Practical09" (download it from FLO).

Complete the definition of the class `Car` by adding

1. instance variables to store the `type` and `model` (both `Strings`) of the car
2. a constructor with the following header:

   ```
   Car(int theCapacity, String theMake, String theType, String theModel)
   ```

   and which assigns the respective instance variables. Note, the first statement in the constructor should be a call to the constructor of the super class (`Vehicle`):

   ```
   super( ...
   ```

The following output should be produced when the program is run:

```
Vehicle Info:
  capacity = 1200
  make = Holden
Vehicle Info:
  capacity = 1500
  make = Mazda
```

Note that the class `Car` does not have a print method and thus the type and model are not printed. This will be added as part of the next task.

Checkpoint 42

Have the program source code and output marked by a demonstrator

Task 2

1. Modify the class Car to include a `print` method which invokes its parent's print method (by making use of `super`) and prints out the type and model.

The following output should be produced when the program is run:

```
Vehicle Info:
  capacity = 1200cc
  make = Holden
  type = sedan
  model = Barina
Vehicle Info:
  capacity = 1500cc
  make = Mazda
  type = sedan
  model = 323
```

Checkpoint 43

Have the program source code and output marked by a demonstrator

Task 3

1. Modify the class `Vehicle`, the base class, to include a method called `setCapacity` which allows the engine capacity to be changed (see example below).
2. Modify the class `Car` so that it overrides the method `setCapacity` with its own version which output the message `"Cannot change capacity of a car"`and does not change the engine capacity. Modify the class `Task` to be:

```
public class Task {

   public static void main(String[] args) {
      Car car1 = new Car(1200,"Holden","sedan","Barina") ;
      Vehicle v1 = new Vehicle(1500,"Mazda") ;
      v1.setCapacity(1600) ;
      v1.print() ;
      car1.setCapacity(1600) ;
      car1.print() ;
   }
}
```

The following output should be produced when the program is run:

```
New capacity = 1600
Vehicle Info:
  capacity = 1600cc
  make = Mazda
Cannot change capacity of a car
Vehicle Info:
  capacity = 1200cc
  make = Holden
  type = sedan
  model = Barina
```

Checkpoint 44

Have the program source code and output marked by a demonstrator

Task 4

1. Add a class called `VehicleDB` which can store up to 100 `Vehicle` objects. It should include the following methods:
   i. a method called addVehicle which adds a Vehicle, or any of its descendants, to the database (see example below).
   ii. a method called print which prints all the Vehicles in the database (see example below). Hint: make use of each Vehicle's print method.
2. Modify the class `Task` to be:

```
public class Task {

   public static void main(String[] args) {
      VehicleDB db = new VehicleDB() ;
      db.addVehicle(new Car(1200,"Holden","sedan","Barina")) ;
      db.addVehicle(new Vehicle(1500,"Mazda")) ;
      db.print() ;
   }
}
```

The following output should be produced when the program is run:

```
=== Vehicle Data Base ===
Vehicle Info:
  capacity = 1200cc
  make = Holden
  type = sedan
  model = Barina
Vehicle Info:
  capacity = 1500cc
  make = Mazda
```

Checkpoint 45

Have the program source code and output marked by a demonstrator

Task 5 (Extension Practice)

1. Add a class called `HoldenDB` which inherits from (extends) `VehicleDB`. It should include a method called `addCar` which takes three parameters (see example below), creates a `Car` object and adds it to the database by making use one of its parent's methods.
2. Modify the class `Task` to be:

```
public class Task {

    public static void main(String[] args) {
        HoldenDB db = new HoldenDB() ;
        db.addCar(1200,"sedan","Barina") ;
        db.addCar(3800,"wagon","Commodore") ;
        db.print() ;
    }
}
```

(Something to think about: do you need to add a constructor to the `HoldenDB` class?)

The following output should be produced when the program is run:

```
=== Vehicle Data Base ===
Vehicle Info:
  capacity = 1200cc
  make = Holden
  type = sedan
  model = Barina
Vehicle Info:
  capacity = 3800cc
  make = Holden
  type = wagon
  model = Commodore
```