OOP Lab Exercise: Inheritance

Submit by mid-day on 24/10/2022

Goals:

- use inheritance to create hierarchies of related classes
- extend behavior and override existing behavior

Inheritance (syntax)

```
public class class name extends superclass {
    ...
}
```

- A subclass *inherits* all of the superclass's behavior and can *override* methods.
- To call an overridden method from the superclass, use the `super` keyword:

```
super.methodName(parameters);
```

Exercise 1: `Car` and `Truck`

```
//Car.java
public class Car {
   public void m1() {
      System.out.println("car 1");
   }

   public void m2() {
      System.out.println("car 2");
   }

   public String toString() {
      return "vroom";
   }
}
```

```
//Truck.java
public class Truck extends Car {
   public void m1() {
      System.out.println("truck 1");
   }
}
```

What is the output from the following code?

```
Truck mycar = new Truck();
System.out.println(mycar);
```

```
mycar.m1();
mycar.m2();
```

Exercise 2: `Car` and `Truck` revisited

```
//Car.java
public class Car {
   public void m1() {
      System.out.println("car 1");
   }

   public void m2() {
      System.out.println("car 2");
   }

   public String toString() {
      return "vroom";
   }
}

//Truck.java
public class Truck extends Car {
   public void m1() {
      System.out.println("truck 1");
   }

   public void m2() {
      super.m1();
   }

   public String toString() {
      return super.toString() + super.toString();
   }
}
```

```
// This client program tests the behavior of your MonsterTruck class.

public class AutoMain {
        public static void main(String[] args) {
                MonsterTruck bigfoot = new MonsterTruck();
                bigfoot.m1();                   // monster 1
                bigfoot.m2();                   // truck 1 / car 1
                System.out.println(bigfoot);    // monster vroomvroom
        }
}
```

Suppose the `Truck` code changes as shown above. What is the output now?

```
Truck mycar = new Truck();
System.out.println(mycar);
mycar.m1();
mycar.m2();
```

Exercise 3: `MonsterTruck`

- Open the following files in an IDE: Car.java, Truck.java, AutoMain.java

Write a class `MonsterTruck` that has the behavior below. Test by running `AutoMain`.

- Some methods produce 2 lines of output; the split between lines is indicated by a `/`.
- Don't just print/return the output; if possible, use inheritance to reuse behavior from the superclass.

```
MonsterTruck bigfoot = new MonsterTruck();
bigfoot.m1();                  // monster 1
bigfoot.m2();                  // truck 1 / car 1
System.out.println(bigfoot);   // monster vroomvroom
```

Exercise 4: `Janitor`

With the help of the following classes: Employee, Secretary, Lawyer, and LegalSecretary.

- **Write a class `Janitor`** to accompany the other employees. Janitors work twice as many hours (80 hours/week), they make $30,000 ($10,000 less than others), they get half as much vacation (only 5 days), and they have an additional method named `clean` that prints "Workin' for the man."
- If you finish, modify it to use the `super` keyword to connect with the `Employee` superclass as appropriate.