Inheritance

CSC 210 Practice Exercises

Concept questions

1. What keyword is used when we want to make use of a superclass in a subclass?
2. What keyword is used to invoke methods of a superclass?
3. What is polymorphism and how is it implemented in Java?
4. When is the keyword `abstract` used?
5. When is the keyword `protected` used?

UML diagrams

Given the UML class diagram below and the application behavior (that shows you the java code to run some code and the output the application produces), write the superclass and subclass in Java.

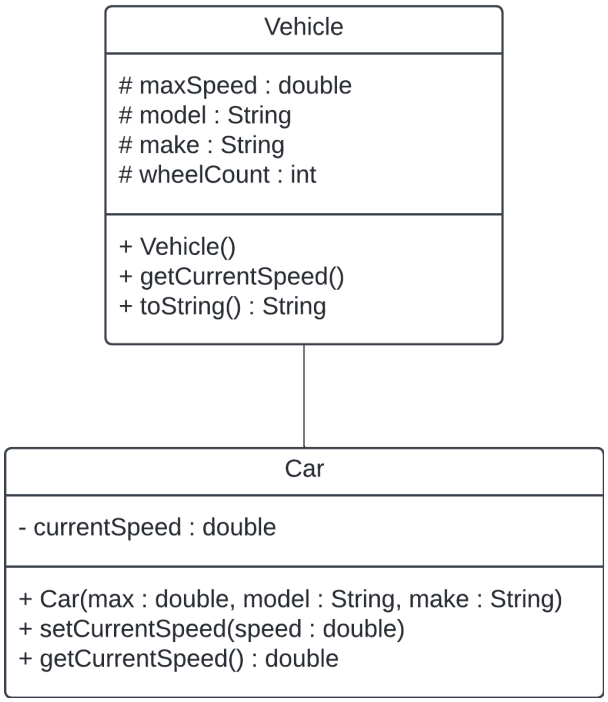
```
public class MyVehicles {

    public static void main(String[] args) {
        Car myCar = new Car(200, "Honda", "Fit");
        myCar.setCurrentSpeed(80);
        System.out.println(myCar.toString());

    }

}
```

```
The vehicle of make Honda and model Fit is going at 80.0 miles/hour
```



Answers

Concept questions

1. `extends`
2. `super`
3. Polymorphism means that one interface can have multiple implementations. Polymorphism can be implemented in java through method overloading (compile-time) and method overriding (runtime).
4. The `abstract` keyword is used to create an abstract superclass, which cannot be instantiated. Instead, the abstract class must be subclassed. Methods in an abstract class can also be abstract – similar to an interface, where the method signature is declared for inheritance. An abstract method is a method that is declared without an implementation.
5. The `protected` keyword is an access modifier (like private, and publick) used for attributes, methods and constructors, making them accessible in subclasses.

UML diagrams

```
public abstract class Vehicle {
    protected double maxSpeed;
    protected String model;
    protected String make;
    protected int wheelCount;

    public Vehicle() {};

    public abstract double getCurrentSpeed();

    public String toString() {
        String message = "";
        message += "The vehicle of make " + make;
        message += " and model " + model;
        message += " is going at " + getCurrentSpeed();
        message += " miles/hour";
        return message;
    }

}
```

```
public class Car extends Vehicle{
    private double currentSpeed;

    public Car(double max, String make, String model) {
        maxSpeed = max;
        this.make = make;
        this.model = model;
        this.wheelCount = 4;
    }

    public void setCurrentSpeed(double speed) {
        currentSpeed = speed;
    }

    public double getCurrentSpeed() {
        return currentSpeed;
    }

}
```