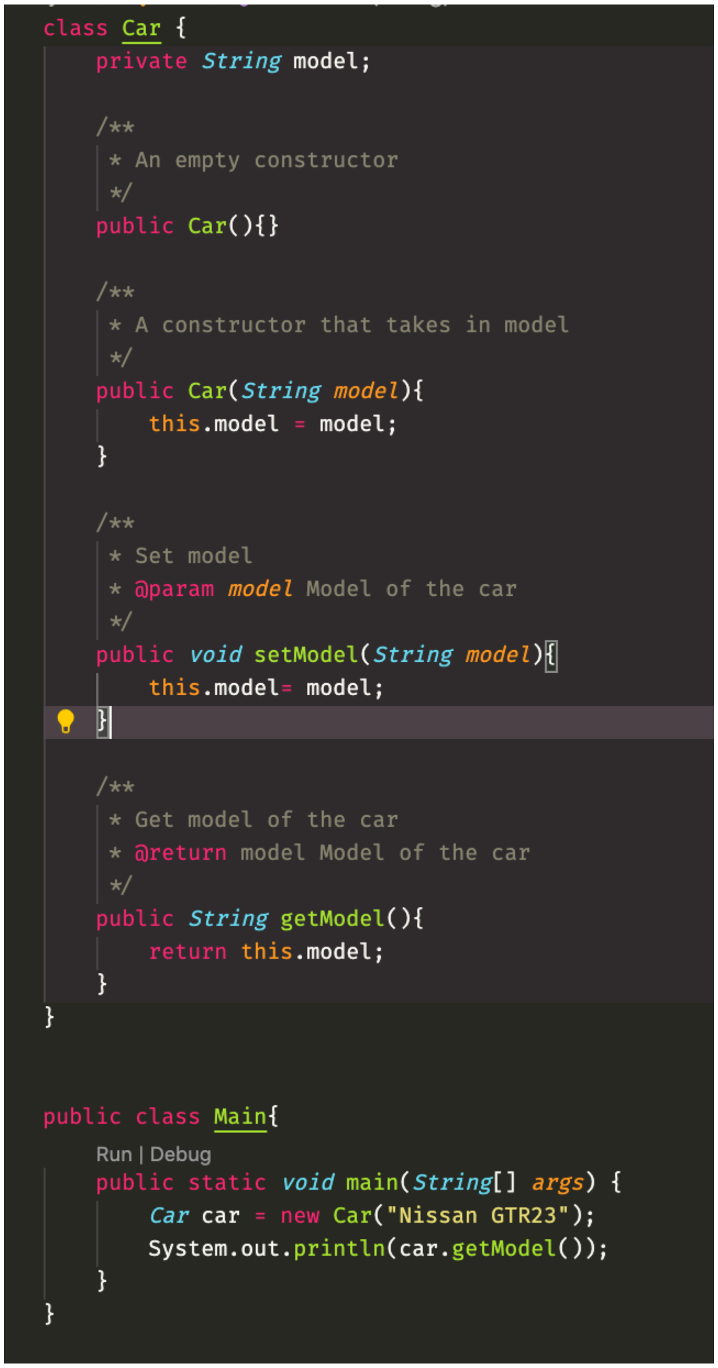
# Object Oriented Concepts

1. Object - Objects are basis to object-oriented programming. Object is anything with a state a behavior in java. In Java, an object is created from a class. In the following example, the variable "car" declared in the main method is an object.



```
class Car {
    private String model;

    /**
     * An empty constructor
     */
    public Car(){}

    /**
     * A constructor that takes in model
     */
    public Car(String model){
        this.model = model;
    }

    /**
     * Set model
     * @param model Model of the car
     */
    public void setModel(String model){
        this.model= model;
    }

    /**
     * Get model of the car
     * @return model Model of the car
     */
    public String getModel(){
        return this.model;
    }
}


public class Main{
    Run | Debug
    public static void main(String[] args) {
        Car car = new Car("Nissan GTR23");
        System.out.println(car.getModel());
    }
}
```

(Picture 1)

1. Class – A class is a blueprint that describes the state and behavior that an object supports. In the previous question, I declared a class called "Car" and gave it a state of "model".
2. **Instantiation of an object –** It means creating a new object. In Java, it's done with "new" keyword. Instantiation of an object has three parts to it.

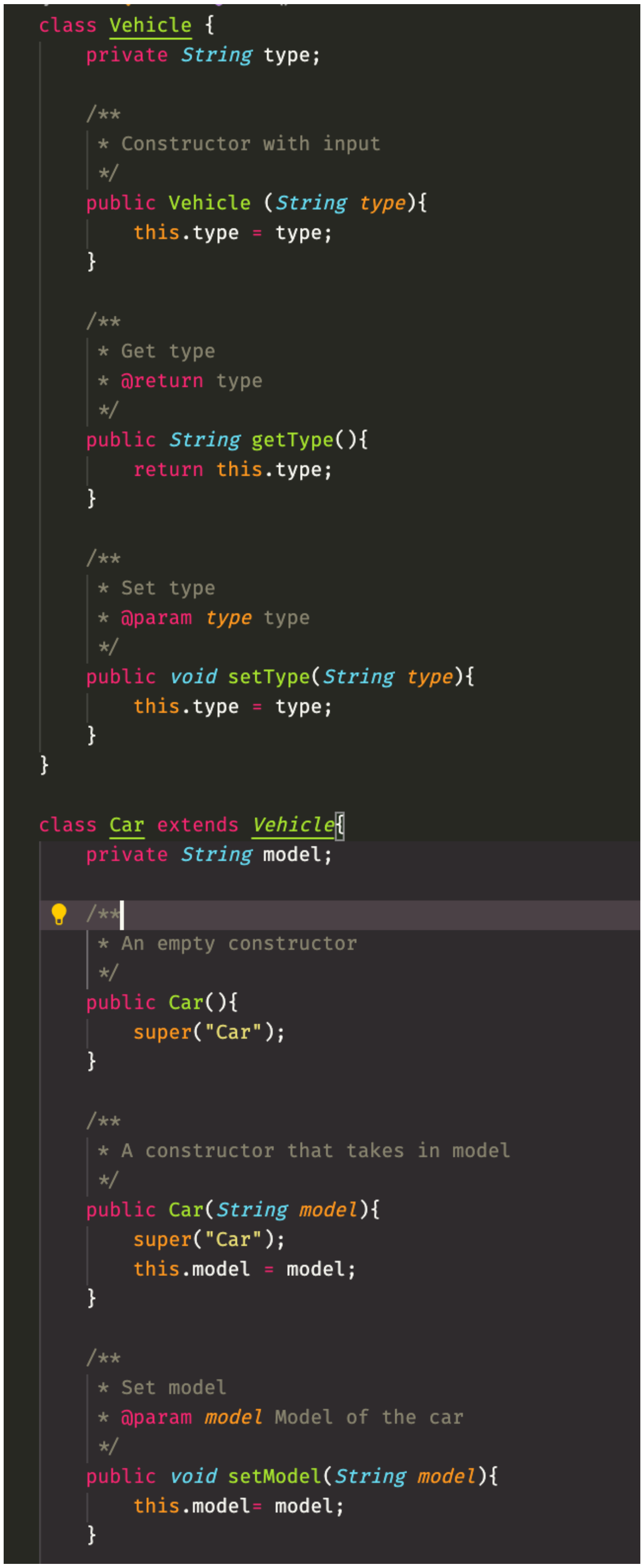
a. Declaration – A variable declaration with a variable name with an object type

b. Instantiation – The keyword "new" comes here

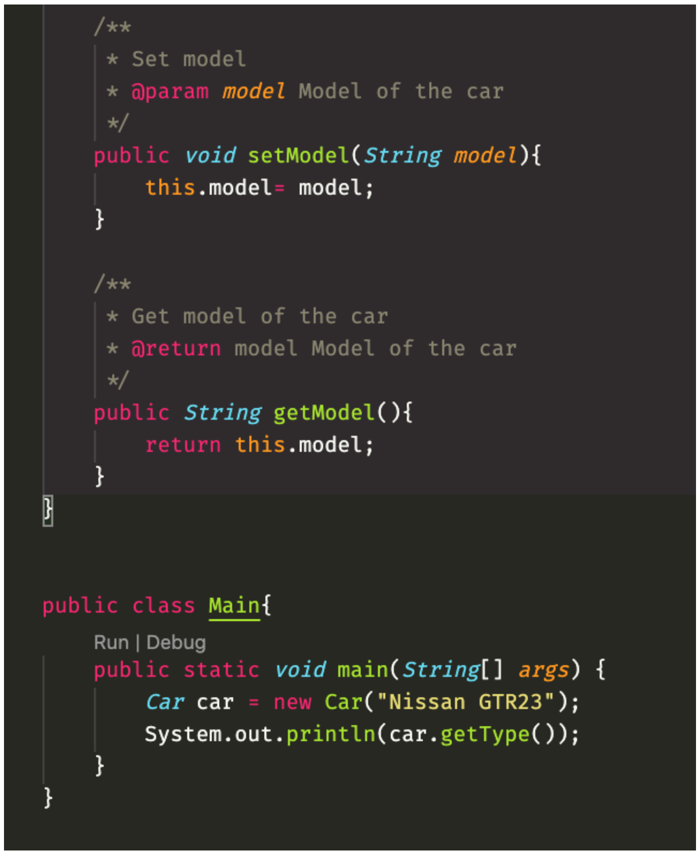
c. Initialization – Then follows a call to the constructor

Referring to Fig1, in the “main” method a new “Car” is instantiated. First is the new variable name of the object, then comes the “new” keyword and then a constructor call of what needs to be instantiated.

3. Visibility – Different visibility levels are set for methods and variables to control access to them. They are set by using the keyword “private”, “public”, “protected” or not specifying it which defaults to “default” Referring to picture 1, we can see how visibility levels are set for both variables and methods. “Private” visibility level is used to stop any access from outside of the class, “protected” limits access to everything but subclasses and other classes in the same package, “public” grants any kind of access and “default” grants access to other classes in the package.
4. Data members and methods – The variables declared inside a class are called data members. They are of two types – Instance and static. Instance data members are those whose memory space is created each time whenever an object is created. Static data members are those whose memory space is created only once, whenever the class is loaded in the main memory irrespective of no of objects are created. Referring to picture1, we can see a data member of type string called “model”. Methods are blocks of code which only run when they are called. They’re declared in classes. Referring to picture1, we can see a method being declared and then used.
5. Inheritance - Inheritance is a mechanism by which an object inherits properties on a parent object. It is accomplished by following the class name where it’s implemented with the keyword “extends” and then name of the class whose properties need to be inherited. This can be seen in the example below.


```
class Vehicle {
    private String type;

    /**
     * Constructor with input
     */
    public Vehicle (String type){
        this.type = type;
    }

    /**
     * Get type
     * @return type
     */
    public String getType(){
        return this.type;
    }

    /**
     * Set type
     * @param type type
     */
    public void setType(String type){
        this.type = type;
    }
}

class Car extends Vehicle{
    private String model;

    /**
     * An empty constructor
     */
    public Car(){
        super("Car");
    }

    /**
     * A constructor that takes in model
     */
    public Car(String model){
        super("Car");
        this.model = model;
    }

    /**
     * Set model
     * @param model Model of the car
     */
    public void setModel(String model){
        this.model= model;
    }
```

(Picture 2)

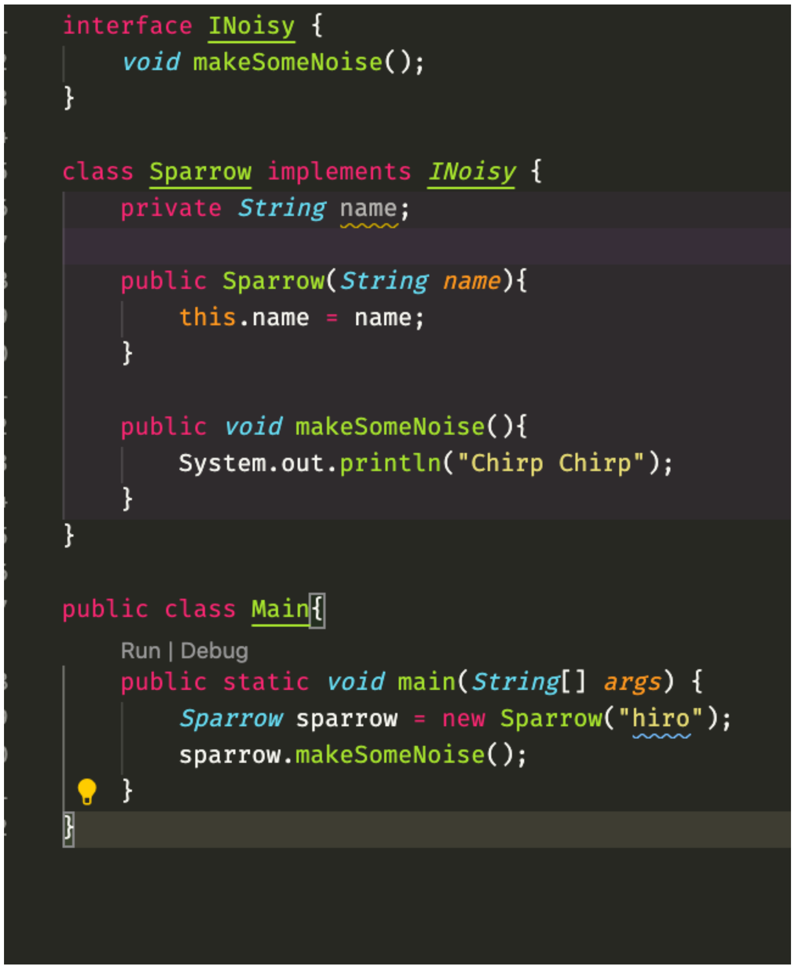


```
    /**
     * Set model
     * @param model Model of the car
     */
    public void setModel(String model){
        this.model= model;
    }

    /**
     * Get model of the car
     * @return model Model of the car
     */
    public String getModel(){
        return this.model;
    }
}


public class Main{
    Run | Debug
    public static void main(String[] args) {
        Car car = new Car("Nissan GTR23");
        System.out.println(car.getType());
    }
}
```

(Picture 3)

Even though the method “getType” is implemented in class “Vehicle”, it’s still usable in class “Car” because class “Car” inherits from class “Vehicle”.

6. Interface – An interface essentially is a class with variables and abstract methods i.e. the methods only have a declaration/signature but no body. It is declared using the keyword “interface” and implemented using the keyword “implements”. Interfaces are used to implement abstraction. Java supports multiple implements from different interfaces.



```
interface INoisy {
    void makeSomeNoise();
}

class Sparrow implements INoisy {
    private String name;

    public Sparrow(String name){
        this.name = name;
    }

    public void makeSomeNoise(){
        System.out.println("Chirp Chirp");
    }
}

public class Main{
    Run | Debug
    public static void main(String[] args) {
        Sparrow sparrow = new Sparrow("hiro");
        sparrow.makeSomeNoise();
    }
}
```

(Picture 4)

7. **Polymorphism –** Polymorphism is the ability to take different forms. For example, code can a class can be extended by other classes. Now the original class is available in several different forms.
8. Overriding - Overriding is overriding prebuilt methods for a specific task. For example, a class may have an overridable method, any class that extends the first class can override any of the overridable methods.
9. Abstract class - A class that does not need instantiation with a constructor.

## More UI

## Android Fundamental Concepts

1. What programming languages you can use for Android app development?
   Java, Kotlin and C++.

2. What is an apk file?

Android SDK took compiles app code along with all its resources into an apk file (Android Package). This file is what gets published on the play where people can find it.

3. How does android system run apps?

4. Name four types of android components? Describe them.

Activities – Activates render the UI and handle user interactions.

Services – They handle background tasks for running the app.

Broadcast Resolvers – They handle communication between android app and android operating system

Content Providers – They provide content and data management.

5. What is manifest file and what is its purpose?

Manifest file contains the data of the android app package including components of the application. It is also responsible for protection of the application from outside entities. It declares rules on how the app is going to interact with the android system. It is saved with the extension "xml".

6. What are resources in android? What are they needed for?

Resources are a major part of all android apps. Apart from code, there's a lot of things that goes into making an android app. They are called resources. Resources are of several types.

**References -**

https://www.oreilly.com/library/view/learning-java-4th/9781449372477/ch06s04.html

http://javaeasy.weebly.com/data-members--methods.html

https://developer.android.com/guide/components/fundamentals

https://www.javatpoint.com/AndroidManifest-xml-file-in-android

https://www.wisdomjobs.com/e-university/android-tutorial-288/what-are-resources-5302.html