Question 01

a) List down the access modifiers and data types that you can use to define the visibility and type of variables and methods in a class.

Access modifiers

- Public
- Private
- Protected

Data types

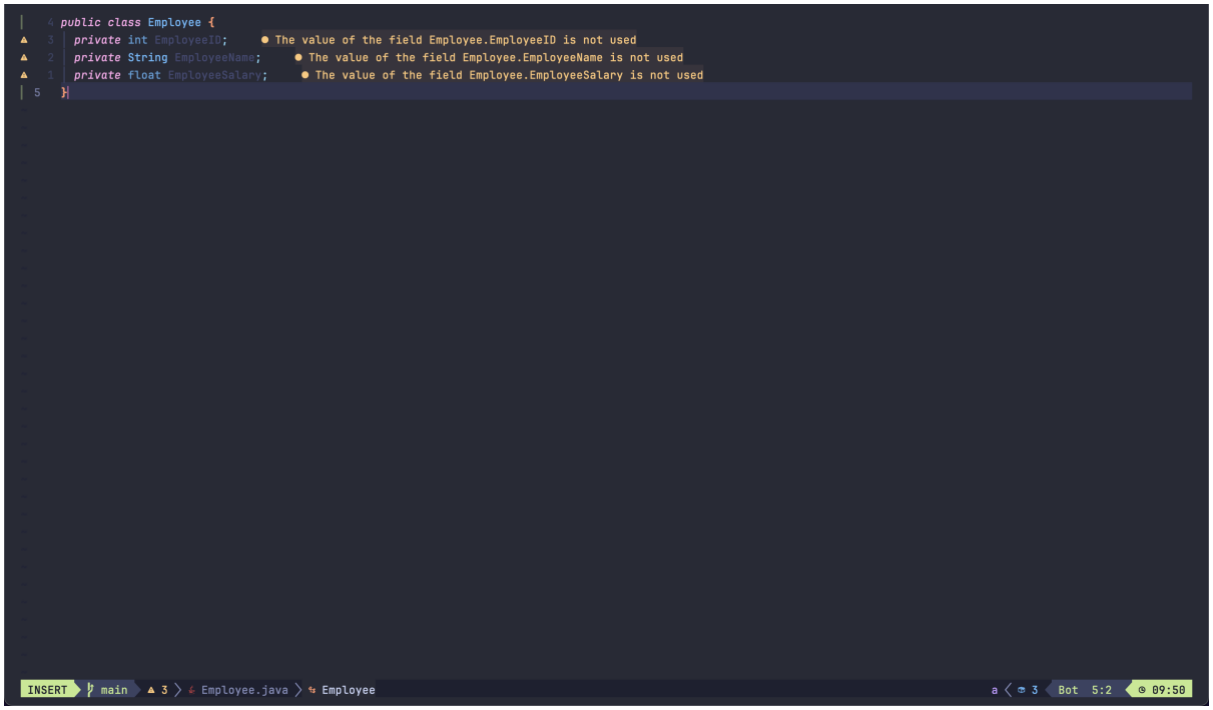
- Int
- String
- Char
- Long
- Float
- Double
- Byte
- Boolean
- Arrays

b) What is the purpose of getter and setter methods in Java?

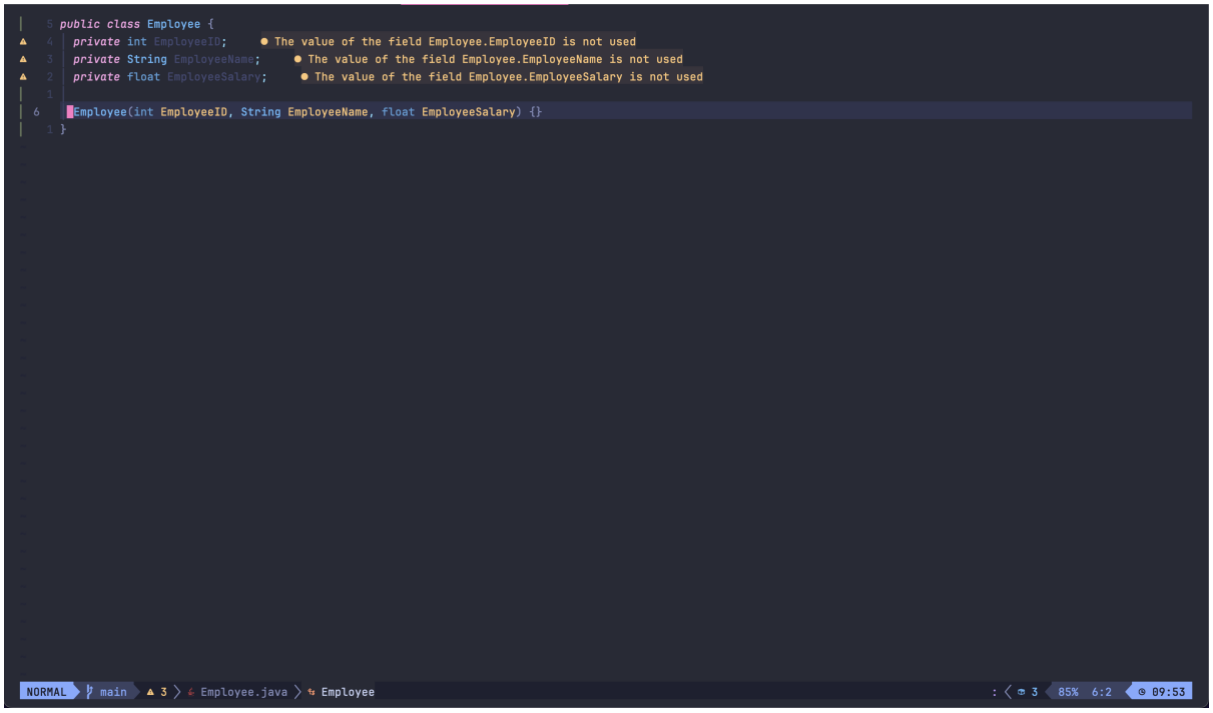
Let's consider a situation where you have a class and have a variable that should not allowed be read then you would generally make it private but making the variable private removed the ability to change the value of the variable this is where the setter method comes in since even though the variable is private we can modify its value from a method within the class we use this ability and create a method called setter that accepts a parameter of the same data type as the private variable and use that method to change the value of the private variable.

Similarly, we use the getter method to read the value of a variable without changing its value in a similar fashion by defining a public method on that class called getter.

c) Create a new class called 'Employee' with three private instance variables, an integer variable called 'EmployeeID', a String variable called 'EmployeeName', and a float variable called "Employee Salary.



d) Add a constructor method for the Employee class that takes the above defined data types as arguments.



e) The constructor should assign the value of these parameters to the corresponding instance variables.

```
public class Employee {
 private int EmployeeID;     ● The value of the field Employee.EmployeeID is not used
 private String EmployeeName;     ● The value of the field Employee.EmployeeName is not used
 private float EmployeeSalary;     ● The value of the field Employee.EmployeeSalary is not used

 Employee(int EmployeeID, String EmployeeName, float EmployeeSalary) {
  this.EmployeeID = EmployeeID;
  this.EmployeeName = EmployeeName;
  this.EmployeeSalary = EmployeeSalary;
 }
}
```

NORMAL main ▲ 3 Employee.java Employee Employee(int, String, float) 3 63% 7:2 09:53

f) Add getter & setter methods for the above three variables.

```
public class Employee {
 private int EmployeeID;
 private String EmployeeName;
 private float EmployeeSalary;

 Employee(int EmployeeID, String EmployeeName, float EmployeeSalary) {
  this.EmployeeID = EmployeeID;
  this.EmployeeName = EmployeeName;
  this.EmployeeSalary = EmployeeSalary;
 }

 public void setEmployeeID(int EmployeeID) {
  this.EmployeeID = EmployeeID;
 }

 public void setEmployeeName(String EmployeeName) {
  this.EmployeeName = EmployeeName;
 }

 public void setEmployeeSalary(float EmployeeSalary) {
  this.EmployeeSalary = EmployeeSalary;
 }

 public int getEmployeeID() {
  return EmployeeID;
 }

 public String getEmployeeName() {
  return EmployeeName;
 }

 public float getEmployeeSalary() {
  return EmployeeSalary;
 }
}
```

NORMAL main Employee.java 3 97% 34:2 09:56

Question 02

You are tasked with implementing a Car class in java, the Car class should contain the following features:

a) Five private instance variables, where three String variables are "make", "mode", and "color", where another integer variable called "year" and the final variable as double variable called "mileage"

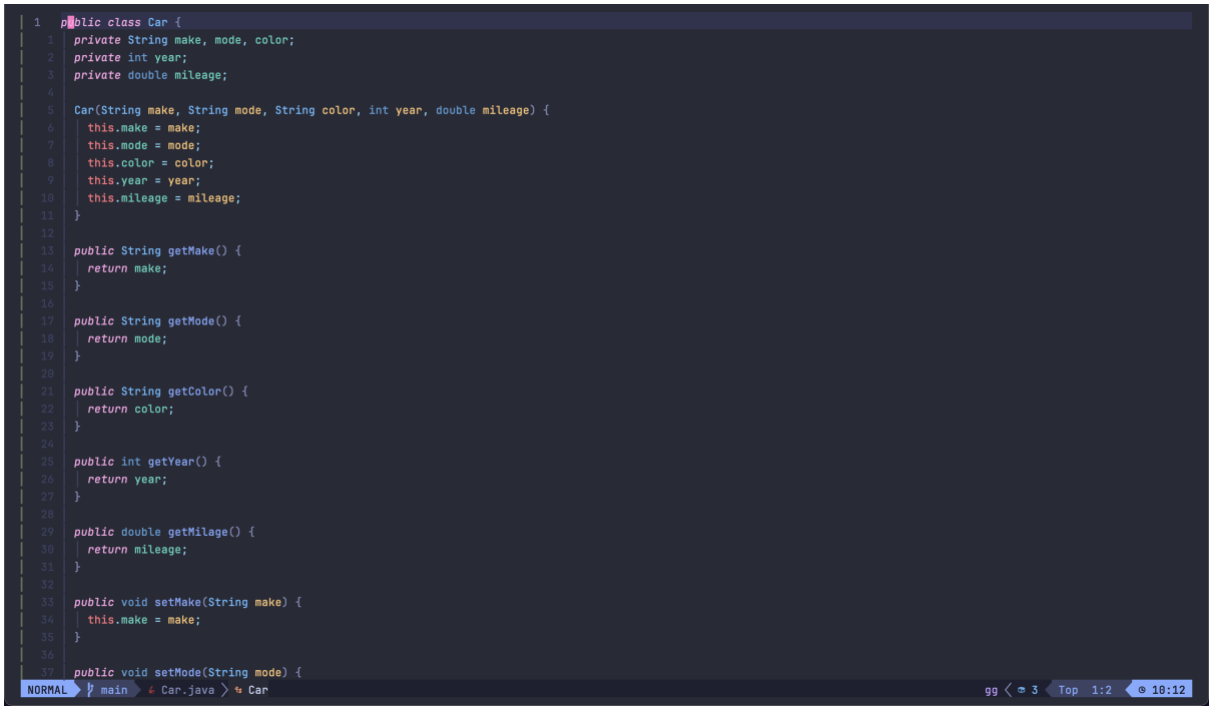
b) A parameterized constructor that takes make,model,color,year and mileage as arguments and initializes the corresponding instance variables.

c) A default constructor that initializes the instance variables with default values.

d) Getter and setter values for all the instance variables

e) A method called drive(distance) that takes a parameter distance (double) and increments the mileage instance variable by given distance.

f) A method called displayCarinfo() that displays the details of the car.



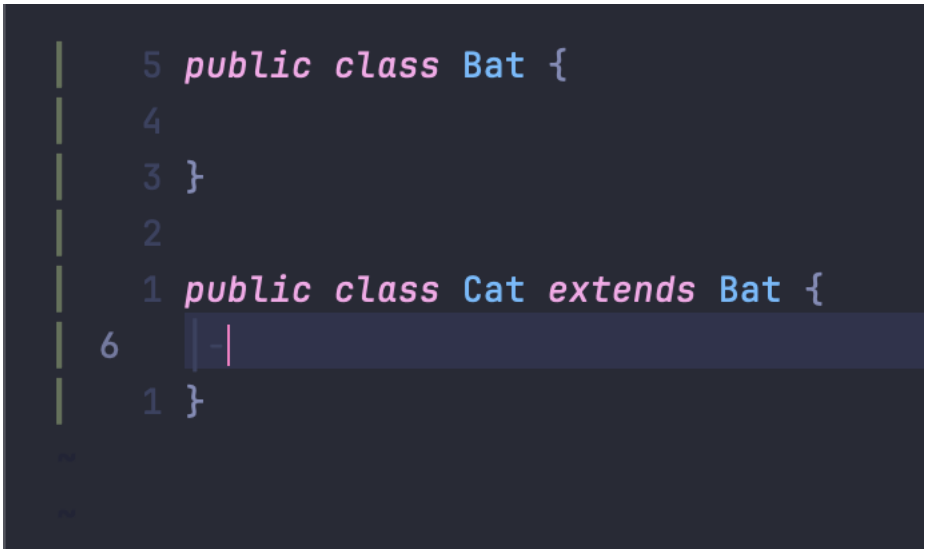
```java
public void setMake(String make) {
  this.make = make;
}

public void setMode(String mode) {
  this.mode = mode;
}

public void setColor(String color) {
  this.color = color;
}

public void setYear(int year) {
  this.year = year;
}

public void setMilage(double mileage) {
  this.mileage = mileage;
}

public void drive(double distance) {
  mileage += distance;
}

public void displayCarInfo() {
  System.out.printf("The make of the car is    : %s", make);
  System.out.printf("The model of the car is   : %s", mode);
  System.out.printf("The color of the car is   : %s", color);
  System.out.printf("The year of the car is    : %d", year);
  System.out.printf("The mileage of the car is : %.2f", mileage);
}
}
```

NORMAL main Car.java Car

Question 03

a) How is Inheritance represented in Java syntax?

Inheritance is the scenario where one class is extends another class for the sake of code simplicity.

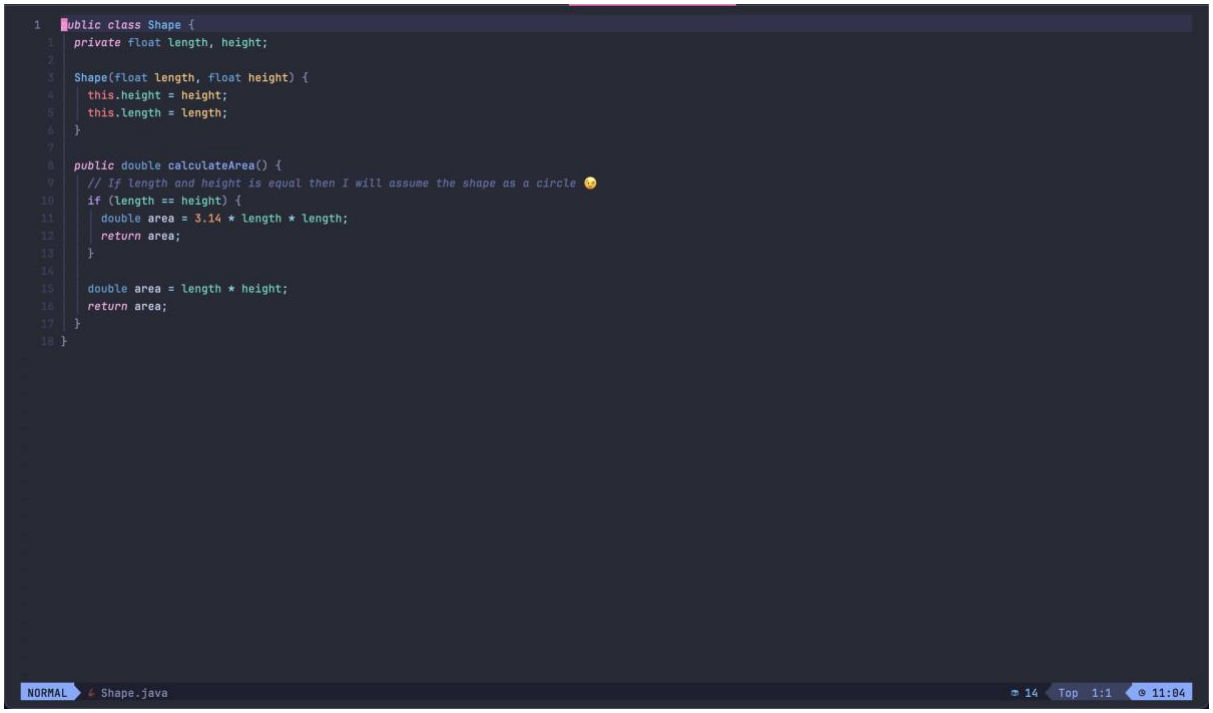


The above diagram represents a basic scenario where the Cat class inherits the Bat class.

According to the above example the Bat class is the super class and the Cat class is the subclass.

b) Create a class called “Shape” with a method called “calculateArea().
c) Create two subclasses , “Circle” and “Rectangle’ , that inherit from the “Shape” class.
d) Implement the calculateArea() method in each subclass to calculate the area of a circle and a rectangle.
e) Test the implementation by creating objects of both subclasses and calling the calculateArea() method on each object.

Shape.java



Circle.java

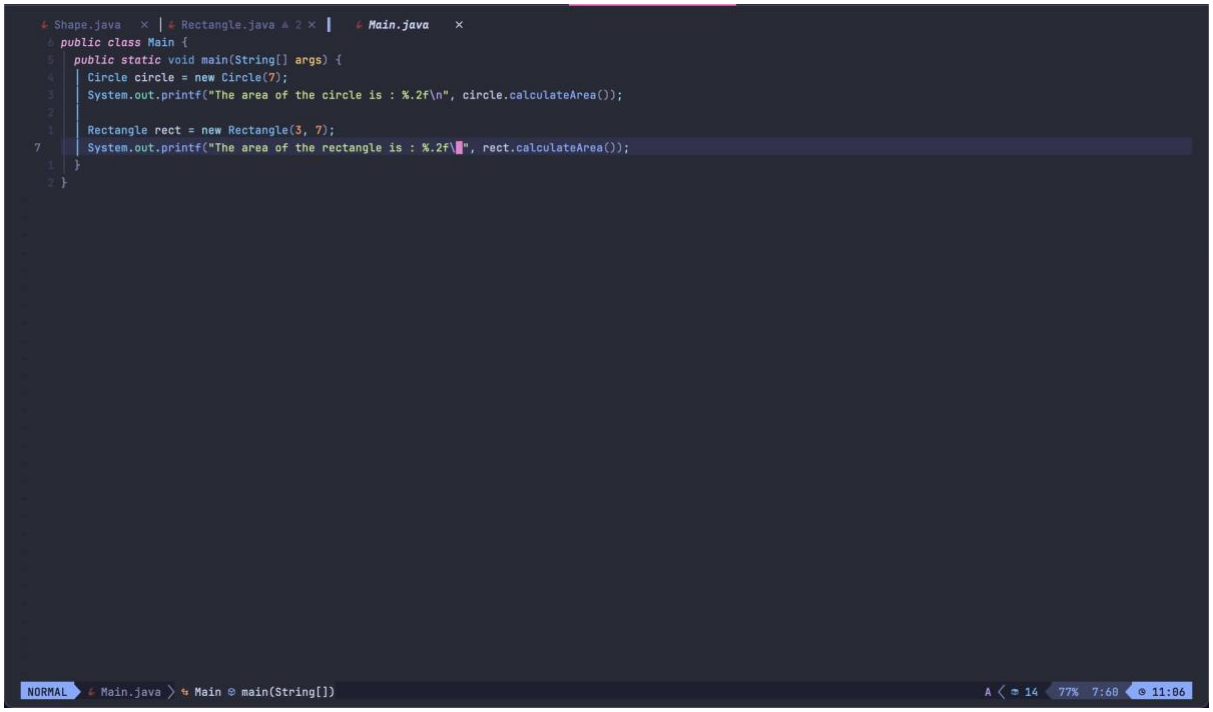
```java
public class Circle extends Shape {
  private float radius;
  Circle(float radius) {
    super(radius, radius);
    this.radius = radius;
  }
}
```

Rectangle.java

```java
public class Rectangle extends Shape {
  private float length, height;
  Rectangle(float length, float height) {
    super(height, length);
    this.height = height;
    this.length = length;
  }
}
```

Main.java

Contains creating objects with the above created classes.


Shape.java
Rectangle.java
Main.java
public class Main {
public static void main(String[] args) {
Circle circle = new Circle(7);
System.out.printf("The area of the circle is : %.2f\n", circle.calculateArea());
Rectangle rect = new Rectangle(3, 7);
System.out.printf("The area of the rectangle is : %.2f\", rect.calculateArea());
}
}
NORMAL
Main.java
Main
main(String[])
14
77%
7:60
11:06