```
}

Encapculation
// Student.java
// Student Class

import java.util.ArrayList;
import java.util.List;

class Student {
  private int student_id;
  private String student_name;
  private List < Double > grades;

  public int getStudent_id() {
    return student_id;
  }

  public void setStudent_id(int student_id) {
    this.student_id = student_id;
  }

  public String getStudent_name() {
    return student_name;
  }

  public void setStudent_name(String student_name) {
    this.student_name = student_name;
  }

  public List < Double > getGrades() {
    return grades;
  }
  public void addGrade(double grade) {
    if (grades == null) {
      grades = new ArrayList < > ();
    }
    grades.add(grade);
  }
}
// Main.java
// Main Class
import java.util.ArrayList;
import java.util.List;
public class Main {
  public static void main(String[] args) {
    // Create an instance of Student
    Student student = new Student();

    // Set the values using the setter methods
    student.setStudent_id(1);
    student.setStudent_name("Nadia Hyakinthos");

    // Add grades using the addGrade() method
    student.addGrade(92.5);
    student.addGrade(89.0);
    student.addGrade(90.3);

    // Get the values using the getter methods
    int student_id = student.getStudent_id();
    String student_name = student.getStudent_name();
    List < Double > grades = student.getGrades();
    // Print the values
    System.out.println("Student ID: " + student_id);
    System.out.println("Student Name: " + student_name);
    System.out.println("Grades: " + grades);
  }
}
```

Polymorphism

```
// Vehicle.java
// Parent class Vehicle

public abstract class Vehicle {
    private String make;
    private String model;
    private int year;
    private String fuelType;
    private double fuelEfficiency;

    public Vehicle(String make, String model, int year, String fuelType, double fuelEfficiency) {
        this.make = make;
        this.model = model;
        this.year = year;
        this.fuelType = fuelType;
        this.fuelEfficiency = fuelEfficiency;
    }

    public String getMake() {
        return make;
    }
    public String getModel() {
        return model;
    }

    public int getYear() {
        return year;
    }

    public String getFuelType() {
        return fuelType;
    }

    public double getFuelEfficiency() {
        return fuelEfficiency;
    }

    public abstract double calculateFuelEfficiency();

    public abstract double calculateDistanceTraveled();

    public abstract double getMaxSpeed();
}
```

Inheritance

```
// Vehicle.java
// Parent class Vehicle

public abstract class Vehicle {
    private String make;
    private String model;
    private int year;

    private String fuelType;

    private double fuelEfficiency;


    public Vehicle(String make, String model, int year, String fuelType, double fuelEfficiency) {

        this.make = make;

        this.model = model;

        this.year = year;

        this.fuelType = fuelType;

        this.fuelEfficiency = fuelEfficiency;

    }


    public String getMake() {

        return make;

    }


    public String getModel() {

        return model;

    }


    public int getYear() {

        return year;

    }


    public String getFuelType() {

        return fuelType;

    }


    public double getFuelEfficiency() {

        return fuelEfficiency;

    }
```

```
    public abstract double calculateFuelEfficiency();

    public abstract double calculateDistanceTraveled();

    public abstract double getMaxSpeed();
}
// Truck.java
// Child class Truck
public class Truck extends Vehicle {
    private double cargoCapacity;

    public Truck(String make, String model, int year, String fuelType, double fuelEfficiency, double cargoCapacity) {
        super(make, model, year, fuelType, fuelEfficiency);
                        //Truck("Ford", "F-150", 2020, "GASOLINE", 8.112);
      this.cargoCapacity = cargoCapacity;
    }

    public double getCargoCapacity() {
        return cargoCapacity;
    }

    @Override
    public double calculateFuelEfficiency() {
        // implementation for fuel efficiency calculation for trucks
        return getFuelEfficiency()*(1.0 / (1.0 + (getCargoCapacity() / 1000.0)));
    }

    @Override
    public double calculateDistanceTraveled() {
        // implementation for distance traveled calculation for trucks
        return calculateFuelEfficiency() * getFuelEfficiency();
    }

    @Override
    public double getMaxSpeed() {
        // implementation for maximum speed calculation for trucks
        return 80.0;
    }
}
// Car.java
// Child class Car

public class Car extends Vehicle {
    private int numSeats;

    public Car(String make, String model, int year, String fuelType, double fuelEfficiency, int numSeats) {
        super(make, model, year, fuelType, fuelEfficiency);
        this.numSeats = numSeats;
    }
    public int getNumSeats() {
        return numSeats;
    }
    @Override
    public double calculateFuelEfficiency() {
        // implementation for fuel efficiency calculation for cars
        return getFuelEfficiency() * (1.0 / (1.0 + (getNumSeats() / 5.0)));
    }
    @Override
    public double calculateDistanceTraveled() {
        // implementation for distance traveled calculation for cars
        return calculateFuelEfficiency() * getFuelEfficiency();
    }

    @Override
    public double getMaxSpeed() {
        // implementation for maximum speed calculation for cars
        return 120.0;
    }
}
// Motorcycle.java
// Child class Motorcycle

public class Motorcycle extends Vehicle {
    private double engineDisplacement;

    public Motorcycle(String make, String model, int year, String fuelType, double fuelEfficiency) {
        super(make, model, year, fuelType, fuelEfficiency);
      //  this.engineDisplacement = engineDisplacement;
    }

    public double getEngineDisplacement() {
        return engineDisplacement;
    }

    @Override
    public double calculateFuelEfficiency() {
        // implementation for fuel efficiency calculation for motorcycles
        return getFuelEfficiency() * (1.0 / (1.0 + (getEngineDisplacement() / 1000.0)));
    }

      @Override
    public double calculateDistanceTraveled() {
        // implementation for distance traveled calculation for cars
        return calculateFuelEfficiency() * getFuelEfficiency();
    }

    @Override
    public double getMaxSpeed() {
        // implementation for maximum speed calculation for cars
        return 80.0;
    }
}
// Main.java
// Main class
public class Main {
    public static void main(String[] args) {

// Create instances of each vehicle type
Truck truck = new Truck("Tatra", "Tatra 810 4x4", 2020, "GASOLINE", 8.112, 4.5);
Car car = new Car("Volkswagen", "Virtus", 2019, "HYBRID", 6.123, 8);
Motorcycle motorcycle = new Motorcycle("Massimo Motor", "Warrior200", 2018, "GASOLINE", 2.1);

// Print the vehicle details and calculations
System.out.println("Truck Model: " + truck.getModel());
System.out.println("Fuel Efficiency: " + truck.calculateFuelEfficiency() + " mpg");
System.out.println("Distance Traveled: " + truck.calculateDistanceTraveled() + " miles");
System.out.println("Max Speed: " + truck.getMaxSpeed() + " mph\n");

System.out.println("Car Model: " + car.getModel());
System.out.println("Fuel Efficiency: " + car.calculateFuelEfficiency() + " mpg");
System.out.println("Distance Traveled: " + car.calculateDistanceTraveled() + " miles");
System.out.println("Max Speed: " + car.getMaxSpeed() + " mph\n");
System.out.println("Motorcycle Model: " + motorcycle.getModel());
System.out.println("Fuel Efficiency: " + motorcycle.calculateFuelEfficiency() + " mpg");
System.out.println("Distance Traveled: " + motorcycle.calculateDistanceTraveled() + " miles");
System.out.println("Max Speed: " + motorcycle.getMaxSpeed() + " mph");
 }
}
```

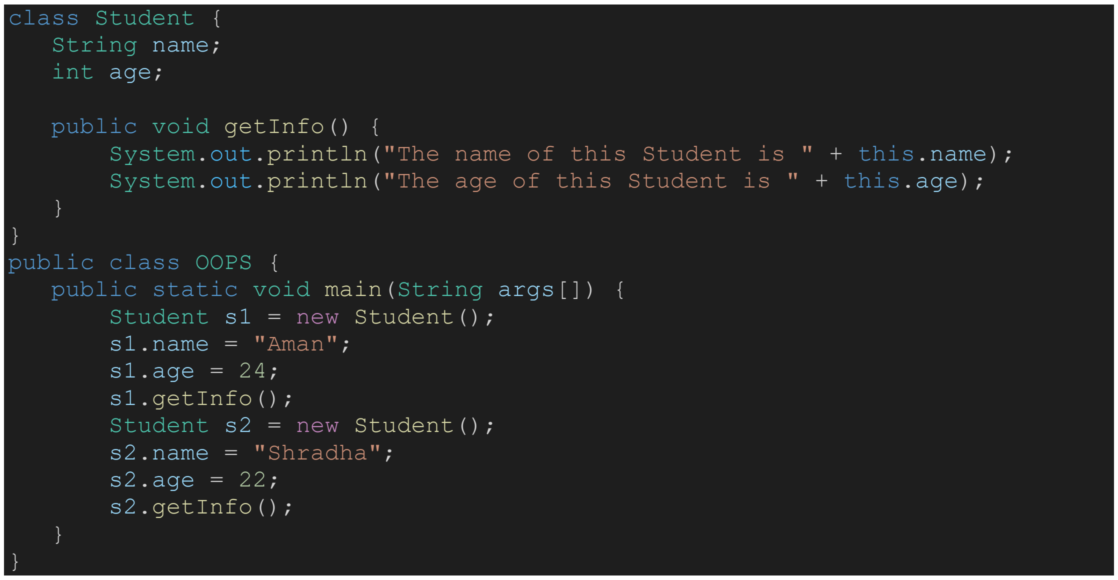
Abstract class

```
//GeometricShape.java
abstract class GeometricShape {
  public abstract double area();

  public abstract double perimeter();
}
//Triangle.java
class Triangle extends GeometricShape {
  private double side1;
  private double side2;
  private double side3;

  public Triangle(double side1, double side2, double side3) {
    this.side1 = side1;
    this.side2 = side2;
    this.side3 = side3;
  }

  @Override
  public double area() {
    double s = (side1 + side2 + side3) / 2;
    return Math.sqrt(s * (s - side1) * (s - side2) * (s - side3));
  }
  @Override
  public double perimeter() {
    return side1 + side2 + side3;
  }
}
//Square.java
class Square extends GeometricShape {
  private double side;

  public Square(double side) {
    this.side = side;
  }

  @Override
  public double area() {
    return side * side;
  }

  @Override
  public double perimeter() {
    return 4 * side;
  }
}
//Main.java
public class Main {
  public static void main(String[] args) {
    GeometricShape triangle = new Triangle(4, 5, 6);
    GeometricShape square = new Square(6);

    System.out.println("Triangle Area: " + triangle.area());
    System.out.println("Triangle Perimeter: " + triangle.perimeter());
    System.out.println("Square Area: " + square.area());
    System.out.println("Square Perimeter: " + square.perimeter());
  }
}
```

Polymorphism

```
//Shape.java
abstract class Shape {
  public abstract void draw();

  public abstract double calculateArea();
}
//Circle.java
class Circle extends Shape {
  private double radius;

  public Circle(double radius) {
    this.radius = radius;
  }

  @Override
  public void draw() {
    System.out.println("Drawing a circle");
  }

  @Override
  public double calculateArea() {
    return Math.PI * radius * radius;
  }
  protected double getRadius() {
    return radius;
  }
}
//Cylinder.java
class Cylinder extends Circle {
  private double height;

  public Cylinder(double radius, double height) {
    super(radius);
    this.height = height;
  }

  @Override
  public void draw() {
    System.out.println("Drawing a cylinder");
  }

  @Override
  public double calculateArea() {
    // Calculate the total surface area of the cylinder (including the circular top and bottom)
    double circleArea = super.calculateArea();
    double sideArea = 2 * Math.PI * getRadius() * height;
    return 2 * circleArea + sideArea;
  }
}
//Main.java
public class Main {
  public static void main(String[] args) {
    Shape circle = new Circle(7.0);
    Shape cylinder = new Cylinder(4.0, 9.0);
    drawShapeAndCalculateArea(circle);
    drawShapeAndCalculateArea(cylinder);
  }

  public static void drawShapeAndCalculateArea(Shape shape) {
    shape.draw();
    double area = shape.calculateArea();
    System.out.println("Area: " + area);
  }
}
```

Class and object



```
class Student {
   String name;
   int age;

   public void getInfo() {
       System.out.println("The name of this Student is " + this.name);
       System.out.println("The age of this Student is " + this.age);
   }
}
public class OOPS {
   public static void main(String args[]) {
       Student s1 = new Student();
       s1.name = "Aman";
       s1.age = 24;
       s1.getInfo();
       Student s2 = new Student();
       s2.name = "Shradha";
       s2.age = 22;
       s2.getInfo();
   }
}
```

```java
class Pen {
   String color;

   public void printColor() {
       System.out.println("The color of this Pen is " + this.color);
   }
}
public class OOPS {
   public static void main(String args[]) {
       Pen p1 = new Pen();
       p1.color = blue;
       Pen p2 = new Pen();
       p2.color = black;
       Pen p3 = new Pen();
       p3.color = red;
       p1.printColor();
       p2.printColor();
       p3.printColor();
   }
}
```

constructor

```java
class Student {
   String name;
   int age;

   Student() {
       System.out.println("Constructor called");
   }
}
```

```java
class Student {
   String name;
   int age;

   Student(String name, int age) {
       this.name = name;
       this.age = age;
   }
}
```

```java
class Student {
   String name;
   int age;

   Student(Student s2) {
       this.name = s2.name;
       this.age = s2.age;
   }
}
```

Function overloading

```
class Student {
   String name;
   int age;

   public void displayInfo(String name) {
       System.out.println(name);
   }
   public void displayInfo(int age) {
       System.out.println(age);
   }
   public void displayInfo(String name, int age) {
       System.out.println(name);
       System.out.println(age);
   }
}
```

Inheritence

```
class Shape {
   public void area() {
       System.out.println("Displays Area of Shape");
   }
}
class Triangle extends Shape {
   public void area(int h, int b) {
       System.out.println((1/2)*b*h);
   }
}
class Circle extends Shape {
   public void area(int r) {
       System.out.println((3.14)*r*r);
   }
}
```

Multilevel Inheritance

```
class Shape {
   public void area() {
       System.out.println("Displays Area of Shape");
   }
}
class Triangle extends Shape {
   public void area(int h, int b) {
       System.out.println((1/2)*b*h);
   }
}
class Circle extends Shape {
   public void area(int r) {
       System.out.println((3.14)*r*r);
   }
}
```

Multiple inheritance

```
class Shape {
   public void area() {
       System.out.println("Displays Area of Shape");
   }
}
class Triangle extends Shape {
   public void area(int h, int b) {
       System.out.println((1/2)*b*h);
   }
}
class EquilateralTriangle extends Triangle {
   int side;
}
```

packages

```
import java.util.Scanner;

import java.io.IOException;
package newpackage;
class Account {
   public String name;
   protected String email;
   private String password;
   public void setPassword(String password) {
       this.password = password;
   }
}
public class Sample {
   public static void main(String args[]) {
       Account a1 = new Account();
       a1.name = "Apna College";
       a1.setPassword("abcd");
       a1.email = "hello@apnacollege.com";
   }
}
```

Abtraction

```
abstract class Animal {
   abstract void walk();
   void breathe() {
       System.out.println("This animal breathes air");
   }
   Animal() {
       System.out.println("You are about to create an Animal.");
   }
}
class Horse extends Animal {
   Horse() {
       System.out.println("Wow, you have created a Horse!");
   }
   void walk() {
       System.out.println("Horse walks on 4 legs");
   }
}
class Chicken extends Animal {
   Chicken() {
       System.out.println("Wow, you have created a Chicken!");
   }
```

```java
    void walk() {
        System.out.println("Chicken walks on 2 legs");
    }
}
public class OOPS {
    public static void main(String args[]) {
        Horse horse = new Horse();
        horse.walk();
        horse.breathe();
    }
}
```

Interface

```java
interface Animal {
    void walk();
}
class Horse implements Animal {
    public void walk() {
        System.out.println("Horse walks on 4 legs");
    }
}
class Chicken implements Animal {
    public void walk() {
        System.out.println("Chicken walks on 2 legs");
    }
}
public class OOPS {
    public static void main(String args[]) {
        Horse horse = new Horse();
        horse.walk();
    }
}
```

Static

```java
class Student {
    static String school;
    String name;
}
public class OOPS {
    public static void main(String args[]) {
        Student.school = "JMV";
        Student s1 = new Student();
        Student s2 = new Student();
        s1.name = "Meena";
        s2.name = "Beena";
        System.out.println(s1.school);
        System.out.println(s2.school);
    }
}
```

```
import javax.swing.*;
import java.io.*;

// Press Shift twice to open the Search Everywhere dialog and type show whitespaces,
// then press Enter. You can now see whitespace characters in your code.
public class Main {
  public static void main(String[] args) throws IOException {
String path="C://Users//zayed//Music//FileHandle//Book1.csv";
    String path2="C://Users//zayed//Music//FileHandle//Book2.csv";

    BufferedReader bufferedReader= new BufferedReader(new FileReader(path));
    BufferedWriter bufferedWriter=new BufferedWriter(new FileWriter(path2));
    String row;
    while((row=bufferedReader.readLine())!=null)
    {
      String[] cell=row.split(",");
      for(String index:cell){
        bufferedWriter.write(index+",");
        System.out.print(index+"\t");



      }
      bufferedWriter.write("\n");
      System.out.print("\n");
    }

bufferedWriter.close();
    bufferedReader.close();

      }
}
```

File read write

```
import javax.swing.*;

import java.io.*;


public class Main {

    public static void main(String[] args) throws IOException {

        String ID="1234";

        String name="Bob";

        String age="22";

        String path="C://Users//zayed//Music//FileHandle//Book1.csv";

        saveRecord(ID, name, age, path);

        showRecord(path);
    }

    public static void saveRecord(String ID, String name, String age, String path) {
        try {
            FileWriter fw = new FileWriter(path, true);
            BufferedWriter bw = new BufferedWriter(fw);
            bw.write(ID + "," + name + "," + age);
            bw.close();
            fw.close();

            JOptionPane.showMessageDialog(null, "Record Saved");
        } catch (Exception e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, "Record not Saved");
        }
    }

    public static void showRecord(String path) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new FileReader(path));
        String row;
        while ((row = bufferedReader.readLine()) != null) {
            String[] cell = row.split(",");
            for (String index : cell) {
                System.out.print(index + "\t");
            }
            System.out.println();
        }
        bufferedReader.close();
    }
}
```

```
package FileOperations;
import java.io.FileWriter;
import java.io.IOException;
public class FileWrite {
public static void main(String[] args) {
try {
FileWriter obj= new FileWriter("/Users/rubyeatislam/eclipse-
workspace/FileInputOuput/src/File2.txt"); obj.write("This is CSE-22.\nThey are
too naughty but responsible. \nCSE-220 class is going on.\n");
obj.close();
System.out.println("Successfully written in file\n");
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} }
}
```

```
package FileOperations;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
public class FileRead {
public static void main(String[] args) {
File obj= new File("/Users/rubyeatislam/eclipse-
workspace/FileInputOuput/src/File2.txt");
try {
Scanner read = new Scanner(obj);
while(read.hasNextLine()) {
String data read.nextLine();
System.out.println(data);
}
} catch (FileNotFoundException e) {
// TODO Auto-generated catch block e.printStackTrace();
}
}
}


abstract class A {
    private int a1;
    protected int a2;
    public String a3;

    A(int x, int y, String a) {
        a1 = x;
        a2 = y;
        a3 = a;
    }

    abstract int fa();

    void display() {
        System.out.println(a1 + " " + a2 + " " + a3);
    }

    int getA1() {
        return a1;
    }

    int getA2() {
        return a2;
    }
}

class B extends A {
    private double b1;

    protected int b2;


    B(int x, int y, String a, double b11, int b22) {

        super(x, y, a);

        b1 = b11;

        b2 = b22;

    }


    int fa() {

        return getA1() + a2 + (int) b1 + b2;

    }


    void display() {

        System.out.println("b1 " + b1 + " b2 " + b2);

    }


    int call_fb() {

        return fb();

    }


    private int fb() {

        return (int) (b1 + b2);

    }


    int ff() {

        return fb();

    }

}


class F extends A {
    protected float f1;

    F(float f11) {
        super(0, 0, "");  // Placeholder values, as A's constructor is not used directly
        f1 = f11;
    }

    int fa() {
        return (int) (f1 + getA1() + a2);
    }

    float getF1() {
        return f1;
    }
}

class G extends B {
    private char g;

    G(int x, int y, String a, double b11, int b22, float f11, char g1) {
        super(x, y, a, b11, b22);
        g = g1;
    }

    int ff() {
        return super.ff() + (int) g;
    }
}

class C extends B {
    private int c1;
    C(int x, int y, String a, double b11, int b22, int c11) {
        super(x, y, a, b11, b22);
        c1 = c11;
    }
}

class D extends B {
    private int d1;

    D(int x, int y, String a, double b11, int b22, int d11) {
        super(x, y, a, b11, b22);
        d1 = d11;
    }
}

class E extends C {
    private int e1;

    E(int x, int y, String a, double b11, int b22, int c11, int d11, int e11) {
        super(x, y, a, b11, b22, c11);
        e1 = e11;
    }
}

public class Main {
    static void display(A obj) {
        obj.display();
    }

    public static void main(String[] args) {
        B B1 = new B(1, 2, "a", 5.2, 2);
        C C1 = new C(1, 2, "a", 5.2, 2, 3);
        G G1 = new G(1, 2, "a", 5.2, 2, 1.9f, 'e');

        int m = G1.ff();
        System.out.println(m);

        A ptr;
        ptr = B1;

        ptr.display();
    }
}
```