**Lab Report**

**Course Tittle:** Object Oriented Programming

**Course Code:** CSE 215

**Experiment No:** 04

**Experiment Name:** Usages of Super Keyword in terms of Inheritance

**Submitted To:**

**Name:** Mst. Umme Ayman

**Designation:** Lecturer

**Department of CSE**

**Daffodil International University**

**Submitted By**

**Name:** Md Raduan Ahamed

**ID:** 0242220005101839

**Section:** 63\_O

**Department of CSE**

**Daffodil International University**

**Submission Date:** 12-09-2023

Problem: 01

*// Define the Student interface*interface Student {

String getName();  
 int getId();  
 String getUniversity();  
 String getDepartment();  
 double getCgpa();  
}  
  
*// Create a class that implements the Student interface*

class UniversityStudent implements Student {  
 private String name;  
 private int id;  
 private String university;  
 private String department;  
 private double cgpa;  
  
 *// Constructor*

public UniversityStudent(String name, int id, String university, String department, double cgpa) {

this.name = name;  
 this.id = id;  
 this.university = university;  
 this.department = department;  
 this.cgpa = cgpa;  
 }  
  
 *// Implement the interface methods*

public String getName() {  
 return name;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public String getUniversity() {  
 return university;  
 }  
  
 public String getDepartment() {  
 return department;  
 }  
  
 public double getCgpa() {  
 return cgpa;  
 }  
  
 @Override  
 public String toString() {

return "Name: " + getName() + "\n" +  
 "ID: " + getId() + "\n" +  
 "University: " + getUniversity() + "\n" +  
 "Department: " + getDepartment() + "\n" +  
 "CGPA: " + getCgpa();  
 }  
}  
  
public class Main {

public static void main(String[] args) {

*// Create a UniversityStudent object*

UniversityStudent student = new UniversityStudent("Md Raduan Ahamed",1839, "Daffodil International University", "Computer Science", 3.75);  
  
 *// Display the student's details* System.out.println("Student Details:");  
 System.out.println(student);  
 }  
}

Output:

**Student Details:**

**Name:** Md Raduan Ahamed

**ID:** 1839

**University:** Daffodil International University

**Department:** Computer Science

**CGPA:** 3.75

Problem: 02

*// Define an interface to represent a geographical entity*

interface GeographicEntity {

double getArea(); *// Method to get the area of the entity* String getName(); *// Method to get the name of the entity*}  
  
*// Create a class to represent a City*

class City implements GeographicEntity {

private String name;  
 private double area;  
  
 public City(String name, double area) {

this.name = name;  
 this.area = area;  
 }  
  
 @Override  
 public double getArea() {

return area;  
 }  
  
 @Override  
 public String getName() {

return name;  
 }  
}  
  
*// Create a class to represent a Village*

class Village implements GeographicEntity {

private String name;  
 private double area;  
  
 public Village(String name, double area) {

this.name = name;  
 this.area = area;  
 }

@Override  
 public double getArea() {

return area;  
 }

@Override  
 public String getName() {

return name;  
 }  
}  
  
*// Create a class to represent a Country*

class Country {

private String name;  
 private long population;  
 private double totalArea;  
 private List<GeographicEntity> entities;  
  
 public Country(String name) {

this.name = name;  
 this.entities = new ArrayList<>();  
 }  
  
 public void addEntity(GeographicEntity entity) {

entities.add(entity);  
 totalArea += entity.getArea();  
 }  
  
 public void setPopulation(long population) {

this.population = population;  
 }  
  
 public long getPopulation() {

return population;  
 }  
  
 public double getTotalArea() {

return totalArea;  
 }  
  
 public String getName() {

return name;  
 }  
  
 public void printCountryInfo() {

System.out.println("Country: " + name);  
 System.out.println("Population: " + population);  
 System.out.println("Total Area: " + totalArea + " square kilometers");  
 System.out.println("Cities and Villages:");  
 for (GeographicEntity entity : entities) {  
 System.out.println("- " + entity.getName() + ": " + entity.getArea() + " square kilometers");

}  
 }  
}

public class Main {

public static void main(String[] args) {

*// Create a country*

Country country = new Country("Bangladesh");  
 country.setPopulation(20,00,00,000);

*// Set the population of the country  
  
 // Create cities and villages*

City city1 = new City("City 1", 300.0);  
 City city2 = new City("City 2", 250.0);  
 Village village1 = new Village("Village 1", 100.0);  
  
 *// Add cities and villages to the country*

country.addEntity(city1);  
 country.addEntity(city2);  
 country.addEntity(village1);  
  
 *// Print country information*

country.printCountryInfo();

}  
}

Output:

**Country:** Bangladesh

**Population:** 20,00,00,000

**Total Area:** 1,47,570.00 square kilometers

**Cities and Villages:**

**- City 1:** 300.0 square kilometers

**- City 2:** 250.0 square kilometers

**- Village 1:** 100.0 square kilometers

Problem: 03

*// Vehicle interface*interface Vehicle {  
 void start();  
 void stop();  
 void honk();  
}  
  
*// Car class implementing Vehicle*

class Car implements Vehicle {  
 private String make;  
 private String model;  
  
 public Car(String make, String model) {  
 this.make = make;  
 this.model = model;  
 }  
  
 @Override  
 public void start() {  
 System.out.println("Starting the " + make + " " + model);  
 }  
  
 @Override  
 public void stop() {  
 System.out.println("Stopping the " + make + " " + model);  
 }  
  
 @Override  
 public void honk() {  
 System.out.println("Honking the horn of the " + make + " " + model);  
 }  
}  
  
*// Bicycle class implementing Vehicle*

class Bicycle implements Vehicle {  
 private String brand;  
  
 public Bicycle(String brand) {  
 this.brand = brand;  
 }  
  
 @Override  
 public void start() {  
 System.out.println("Starting the " + brand + " bicycle");  
 }  
  
 @Override  
 public void stop() {  
 System.out.println("Stopping the " + brand + " bicycle");  
 }  
  
 @Override  
 public void honk() {  
 System.out.println("Bicycles don't have horns!");  
 }  
}  
  
public class Main {  
 public static void main(String[] args) {  
 Vehicle car = new Car("Toyota", "Camry");  
 Vehicle bicycle = new Bicycle("Trek");  
  
 car.start();  
 car.honk();  
 car.stop();  
  
 System.out.println();  
  
 bicycle.start();  
 bicycle.honk();  
 bicycle.stop();  
 }  
}

Output:

Starting the Toyota Camry

Honking the horn of the Toyota Camry

Stopping the Toyota Camry

Starting the Trek bicycle

Bicycles don't have horns!

Stopping the Trek bicycle