ASSIGNMENT 5 – DCIT 201 (PROGRAMMING 1)

1128 3519

1. **QUESTION NUMBER 1**

```
public class Car {
    private double fuelEfficiency; // Miles per gallon
    private double fuelLevel; // Current fuel level in gallons

    // Constructor to initialize the fuel efficiency and set initial fuel level to 0
    public Car(double fuelEfficiency) {
        this.fuelEfficiency = fuelEfficiency;
        this.fuelLevel = 0;
    }

    // Method to simulate driving the car for a certain distance
    public void drive(double distance) {
        // Calculate the fuel consumed during the drive
        double fuelConsumed = distance / fuelEfficiency;

        // Check if there is enough fuel to complete the drive
        if (fuelConsumed <= fuelLevel) {
            fuelLevel -= fuelConsumed;
        } else {
            // If there's not enough fuel, drive as far as possible
            fuelLevel = 0;
        }
    }

    // Method to return the current fuel level
    public double getGasLevel() {
```

```
        return fuelLevel;
    }

    // Method to add fuel to the gas tank
    public void addGas(double gallons) {
        if (gallons > 0) {
            fuelLevel += gallons;
        }
    }

    public static void main(String[] args) {
        Car myHybrid = new Car(50); // 50 miles per gallon
        myHybrid.addGas(20); // Tank up with 20 gallons
        myHybrid.drive(100); // Drive 100 miles

        // Print the remaining fuel
        System.out.println("Fuel remaining: " + myHybrid.getGasLevel() + " gallons");
    }
}
```

2. **QUESTION NUMBER 2**

```
public class Student {
   private String name;
   private int totalQuizScore;
   private int numberOfQuizzes;

   // Constructor to initialize the student's name and quiz-related variables
   public Student(String name) {
      this.name = name;
      this.totalQuizScore = 0;
      this.numberOfQuizzes = 0;
   }

   // Method to get the student's name
   public String getName() {
      return name;
   }

   // Method to add a quiz score and update the total score and number of quizzes
   public void addQuiz(int score) {
      totalQuizScore += score;
      numberOfQuizzes++;
   }

   // Method to get the total quiz score
   public int getTotalScore() {
      return totalQuizScore;
   }

   // Method to calculate and get the average quiz score
   public double getAverageScore() {
      if (numberOfQuizzes > 0) {
         return (double) totalQuizScore / numberOfQuizzes;
      } else {
         return 0.0; // To avoid division by zero if no quizzes have been taken
      }
   }

   public static void main(String[] args) {
      // Create a Student instance
      Student student = new Student("John");

      // Add quiz scores
      student.addQuiz(80);
      student.addQuiz(90);
```

```
        student.addQuiz(75);

        // Display student information
        System.out.println("Student Name: " + student.getName());
        System.out.println("Total Quiz Score: " + student.getTotalScore());
        System.out.println("Average Quiz Score: " + student.getAverageScore());
    }
}
```

3. **QUESTION NUMBER 3**

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

class Country {
   private String name;
   private long population;
   private double area;

   public Country(String name, long population, double area) {
      this.name = name;
      this.population = population;
      this.area = area;
   }

   public String getName() {
      return name;
   }

   public long getPopulation() {
      return population;
   }

   public double getArea() {
      return area;
   }

   public double getPopulationDensity() {
```

```
        return population / area;
    }
}

public class CountryAnalyzer {
    public static void main(String[] args) {
        List<Country> countries = new ArrayList<>();

        // Add some example countries
        countries.add(new Country("USA", 331000000, 9833520));
        countries.add(new Country("Canada", 38000000, 9984670));
        countries.add(new Country("China", 1444216107, 9596961));
        countries.add(new Country("India", 1380004385, 3287263));
        countries.add(new Country("Brazil", 212559417, 8515767));

        // Find the country with the largest area
        Country largestAreaCountry = Collections.max(countries,
Comparator.comparing(Country::getArea));

        // Find the country with the largest population
        Country largestPopulationCountry = Collections.max(countries,
Comparator.comparing(Country::getPopulation));

        // Find the country with the largest population density
        Country largestPopulationDensityCountry = Collections.max(countries,
Comparator.comparing(Country::getPopulationDensity));

        // Print the results
        System.out.println("Country with the largest area: " + largestAreaCountry.getName());
        System.out.println("Country with the largest population: " +
largestPopulationCountry.getName());
```

```
        System.out.println("Country with the largest population density: " +
largestPopulationDensityCountry.getName());

    }

}
```

4. **QUESTION NUMBER 4**

```
class Person {
   private String name;
   private int yearOfBirth;

   public Person(String name, int yearOfBirth) {
      this.name = name;
      this.yearOfBirth = yearOfBirth;
   }

   public String getName() {
      return name;
   }

   public int getYearOfBirth() {
      return yearOfBirth;
   }
}

class Student extends Person {
   private String major;

   public Student(String name, int yearOfBirth, String major) {
      super(name, yearOfBirth);
      this.major = major;
   }

   public String getMajor() {
      return major;
   }
}

class Instructor extends Person {
   private double salary;

   public Instructor(String name, int yearOfBirth, double salary) {
      super(name, yearOfBirth);
      this.salary = salary;
   }

   public double getSalary() {
      return salary;
   }
}
```

```
public class TestPerson {
   public static void main(String[] args) {
      Student student = new Student("Alice", 2000, "Computer Science");
      Instructor instructor = new Instructor("Bob", 1985, 60000.0);

      System.out.println("Student Information:");
      System.out.println("Name: " + student.getName());
      System.out.println("Year of Birth: " + student.getYearOfBirth());
      System.out.println("Major: " + student.getMajor());

      System.out.println();

      System.out.println("Instructor Information:");
      System.out.println("Name: " + instructor.getName());
      System.out.println("Year of Birth: " + instructor.getYearOfBirth());
      System.out.println("Salary: $" + instructor.getSalary());
   }
}
```

5. **QUESTION NUMBER 5**

```
class Employee {
    private String name;
    private double salary;

    public Employee(String name, double salary) {
        this.name = name;
        this.salary = salary;
    }

    @Override
    public String toString() {
        return "Employee Name: " + name + "\nSalary: $" + salary;
    }
}

class Manager extends Employee {
    private String department;

    public Manager(String name, double salary, String department) {
        super(name, salary);
        this.department = department;
    }

    @Override
    public String toString() {
        return "Manager Name: " + getName() + "\nDepartment: " + department + "\nSalary: $" +
getSalary();
    }
}
```

```
class Executive extends Manager {
    public Executive(String name, double salary, String department) {
        super(name, salary, department);
    }

    @Override
    public String toString() {
        return "Executive Name: " + getName() + "\nDepartment: " + getDepartment() + "\nSalary:
$" + getSalary();
    }
}

public class TestEmployee {
    public static void main(String[] args) {
        Employee employee = new Employee("John Doe", 50000.0);
        Manager manager = new Manager("Alice Smith", 75000.0, "Marketing");
        Executive executive = new Executive("Bob Johnson", 100000.0, "Finance");

        System.out.println("Employee Information:");
        System.out.println(employee);

        System.out.println("\nManager Information:");
        System.out.println(manager);

        System.out.println("\nExecutive Information:");
        System.out.println(executive);
    }
}
```