Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

1. Write a Java Program that implements Multiple inheritance concept based on example of student faculty template?

```
// Interface for the Student
interface Student {
    void study();
    void attendClasses();
}

// Interface for the Faculty
interface Faculty {
    void teach();
    void conductExams();
}

// Class implementing both Student and Faculty interfaces
class StudentFaculty implements Student, Faculty {

    // Student methods
    @Override
    public void study() {
        System.out.println("Student is studying");
    }

    @Override
    public void attendClasses() {
        System.out.println("Student is attending classes");
    }

    // Faculty methods
    @Override
    public void teach() {
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

```
        System.out.println("Faculty is teaching");
    }

    @Override
    public void conductExams() {
        System.out.println("Faculty is conducting exams");
    }

    // Additional methods specific to StudentFaculty
    public void participateInActivities() {
        System.out.println("StudentFaculty is participating in activities");
    }
}

// Main class to test the implementation
public class Main {
    public static void main(String[] args) {
        // Create an instance of StudentFaculty
        StudentFaculty studentFaculty = new StudentFaculty();

        // Call methods from both Student and Faculty interfaces
        studentFaculty.study();
        studentFaculty.attendClasses();
        studentFaculty.teach();
        studentFaculty.conductExams();

        // Call additional method from the StudentFaculty class
        studentFaculty.participateInActivities();
    }
}
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

Output-

Student is studying

Student is attending classes

Faculty is teaching

Faculty is conducting exams

StudentFaculty is participating in activities

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

2. Write a program to print the names of students by creating a Student class. If no name is passed while creating an object of Student class, then the name should be "Unknown", otherwise the name should be equal to the String value passed while creating object of Student class.

```
class Student {
    private String name;

    // Constructor with a default name of "Unknown"
    public Student() {
        this.name = "Unknown";
    }

    // Constructor with a parameter to set the name
    public Student(String name) {
        this.name = name;
    }

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

public class Main {
    public static void main(String[] args) {
        // Creating a Student with no name specified
        Student student1 = new Student();
        System.out.println("Student 1 Name: " + student1.getName());

        // Creating a Student with a specific name
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

```
        Student student2 = new Student("John Doe");
        System.out.println("Student 2 Name: " + student2.getName());
    }
}
```

Output-

Student 1 Name: Unknown

Student 2 Name: John Doe

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

3.Write a Java program to get the character (Unicode code point) at the given index within the string

```
import java.util.Scanner;


public class UnicodeAtIndex {
    public static void main(String[] args) {
        // Create a Scanner object for user input
        Scanner scanner = new Scanner(System.in);

        // Get the input string from the user
        System.out.print("Enter a string: ");
        String inputString = scanner.nextLine();

        // Get the index from the user
        System.out.print("Enter the index: ");
        int index = scanner.nextInt();

        // Check if the index is within the valid range
        if (index >= 0 && index < inputString.length()) {
            // Get the Unicode code point of the character at the specified index
            int unicodeCodePoint = inputString.codePointAt(index);

            // Print the result
            System.out.println("Unicode code point at index " + index + ": " + unicodeCodePoint);
        } else {
            System.out.println("Invalid index. Please enter a valid index within the string length.");
        }

        // Close the Scanner
        scanner.close();
    }
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

```
}
```

Output-

```
Enter a string: Hello, World!
Enter the index: 7
Unicode code point at index 7: 87
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

4. Create an abstract class 'Parent' with a method 'message'. It has two subclasses each having a method with the same name 'message' that prints "This is first subclass" and "This is second subclass" respectively. Call the methods 'message' by creating an object for each subclass.

```
// Abstract class 'Parent'
abstract class Parent {
    // Abstract method 'message'
    abstract void message();
}

// First subclass
class FirstSubclass extends Parent {
    // Implementation of the 'message' method for the first subclass
    @Override
    void message() {
        System.out.println("This is first subclass");
    }
}

// Second subclass
class SecondSubclass extends Parent {
    // Implementation of the 'message' method for the second subclass
    @Override
    void message() {
        System.out.println("This is second subclass");
    }
}

public class Main {
    public static void main(String[] args) {
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

```
        // Create an object for the first subclass
        FirstSubclass firstObject = new FirstSubclass();
        // Call the 'message' method for the first subclass
        firstObject.message();

        // Create an object for the second subclass
        SecondSubclass secondObject = new SecondSubclass();
        // Call the 'message' method for the second subclass
        secondObject.message();
    }
}
```

Output-

```
This is first subclass
This is second subclass
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

5.WAP that show the partial implementation of interface

```
// Interface with two methods
interface MyInterface {
    void method1();  // Abstract method
    void method2();  // Abstract method
}

// Concrete class implementing the interface partially
class MyPartialImplementation implements MyInterface {

    // Implementing only method1 from the interface
    @Override
    public void method1() {
        System.out.println("Implementation of method1");
    }

    // No implementation for method2 in this class

    // Additional method specific to this class
    public void additionalMethod() {
        System.out.println("Additional method in MyPartialImplementation");
    }
}

public class Main {
    public static void main(String[] args) {
        // Create an object of MyPartialImplementation
        MyPartialImplementation myObj = new MyPartialImplementation();

        // Call the implemented method from the interface
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

```
        myObj.method1();

        // Call the additional method specific to MyPartialImplementation
        myObj.additionalMethod();

        // Since method2 is not implemented in this class, it cannot be called directly.
        // Uncommenting the line below will result in a compilation error:
        // myObj.method2();
    }
}
```

Output-

Implementation of method1

Additional method in MyPartialImplementation

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

6. Write a Java program to create an array list, add some colors (strings) and print out the collection.

```
import java.util.ArrayList;


public class ColorArrayList {
    public static void main(String[] args) {
        // Create an ArrayList to store colors
        ArrayList<String> colorList = new ArrayList<>();

        // Add some colors to the ArrayList
        colorList.add("Red");
        colorList.add("Green");
        colorList.add("Blue");
        colorList.add("Yellow");

        // Print out the collection
        System.out.println("Colors in the ArrayList:");

        // Using for-each loop to iterate and print each color
        for (String color : colorList) {
            System.out.println(color);
        }
    }
}
```

Output-

```
Colors in the ArrayList:
Red
Green
Blue
Yellow
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

7.Write a Java program to insert elements into the linked list at the first and last positions.

```
import java.util.LinkedList;


public class LinkedListExample {
    public static void main(String[] args) {
        // Create a LinkedList to store elements
        LinkedList<String> linkedList = new LinkedList<>();

        // Insert elements at the first and last positions
        linkedList.addFirst("Element at the Beginning");
        linkedList.addLast("Element at the End");

        // Print out the linked list
        System.out.println("Linked List elements:");

        // Using for-each loop to iterate and print each element
        for (String element : linkedList) {
            System.out.println(element);
        }
    }
}
```

Output-

```
 Linked List elements:
Element at the Beginning
Element at the End
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

8.Write a Java program to convert a hash set to an array.

```
import java.util.HashSet;
import java.util.Arrays;

public class HashSetToArray {
    public static void main(String[] args) {
        // Create a HashSet
        HashSet<String> hashSet = new HashSet<>();

        // Add elements to the HashSet
        hashSet.add("Apple");
        hashSet.add("Banana");
        hashSet.add("Orange");
        hashSet.add("Grapes");
        // Convert HashSet to an array
        String[] array = new String[hashSet.size()];
        hashSet.toArray(array);
        // Print out the array
        System.out.println("Array elements:");
        // Using for-each loop to iterate and print each element in the array
        for (String element : array) {
            System.out.println(element);
        }
    }
}
```

Output- Array elements:

Grapes

Orange

Apple

Banana

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

9. Write a Java program to check whether a given string ends with another string.

```
import java.util.Scanner;

public class CheckEndsWith {
    public static void main(String[] args) {
        // Create a Scanner object for user input
        Scanner scanner = new Scanner(System.in);

        // Get the main string from the user
        System.out.print("Enter the main string: ");
        String mainString = scanner.nextLine();

        // Get the suffix string from the user
        System.out.print("Enter the suffix to check: ");
        String suffix = scanner.nextLine();

        // Check if the main string ends with the specified suffix
        boolean endsWith = mainString.endsWith(suffix);

        // Print the result
        if (endsWith) {
            System.out.println("The main string ends with the specified suffix.");
        } else {
            System.out.println("The main string does not end with the specified suffix.");
        }

        // Close the Scanner
        scanner.close();
    }
}
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

Output-

Enter the main string: Hello, World!

Enter the suffix to check: World!

The main string ends with the specified suffix.

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

10. Write a Java program to remove the third element from an array list.

```
import java.util.ArrayList;
public class RemoveElement {
    public static void main(String[] args) {
        // Create an ArrayList
        ArrayList<String> arrayList = new ArrayList<>();
        // Add elements to the ArrayList
        arrayList.add("Element 1");
        arrayList.add("Element 2");
        arrayList.add("Element 3");
        arrayList.add("Element 4");
        // Print the original ArrayList
        System.out.println("Original ArrayList: " + arrayList);
        // Remove the third element (index 2)
        if (arrayList.size() >= 3) {
            arrayList.remove(2);
            System.out.println("Element at index 2 removed.");
        } else {
            System.out.println("ArrayList does not have a third element to remove.");
        }
        // Print the ArrayList after removal
        System.out.println("ArrayList after removal: " + arrayList);
    }
}
```

Output-

```
Original ArrayList: [Element 1, Element 2, Element 3, Element 4]
Element at index 2 removed.
ArrayList after removal: [Element 1, Element 2, Element 4]
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

11. Perform hibernate crud operation

Hibernate Configration file-

```
<!-- hibernate.cfg.xml -->
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">


<hibernate-configuration>
   <session-factory>
        <!-- JDBC Database connection settings -->
        <property
name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/your_database_name</propert
y>
        <property name="hibernate.connection.username">your_username</property>
        <property name="hibernate.connection.password">your_password</property>
        <!-- JDBC connection pool settings -->
        <property name="hibernate.c3p0.min_size">5</property>
        <property name="hibernate.c3p0.max_size">20</property>
        <property name="hibernate.c3p0.timeout">300</property>
        <property name="hibernate.c3p0.max_statements">50</property>
        <property name="hibernate.c3p0.idle_test_period">3000</property>
        <!-- Specify dialect -->
        <property name="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</property>
        <!-- Echo all executed SQL to stdout -->
        <property name="hibernate.show_sql">true</property>
        <!-- Drop and re-create the database schema on startup -->
        <property name="hibernate.hbm2ddl.auto">update</property>
        <!-- Mention annotated class -->
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

```
        <mapping class="com.example.model.Student"/>
    </session-factory>
</hibernate-configuration>
```

Entity Class-

```
// Student.java
import javax.persistence.*;

@Entity
@Table(name = "students")
public class Student {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;

    @Column(name = "name")
    private String name;

    @Column(name = "age")
    private int age;

    // Constructors, getters, and setters
}
```

Hibernate Crud operation-

```
// HibernateCRUDExample.java
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

```
public class HibernateCRUDExample {
    public static void main(String[] args) {
        // Create a SessionFactory
        SessionFactory factory = new Configuration().configure("hibernate.cfg.xml")
                .addAnnotatedClass(Student.class)
                .buildSessionFactory();

        // Create a Session
        Session session = factory.getCurrentSession();

        try {
            // Begin transaction
            session.beginTransaction();

            // Perform CRUD operations

            // Create
            Student student = new Student("John Doe", 25);
            session.save(student);
            System.out.println("Student saved. ID: " + student.getId());

            // Read
            Student retrievedStudent = session.get(Student.class, student.getId());
            System.out.println("Retrieved Student: " + retrievedStudent);

            // Update
            retrievedStudent.setName("Updated Name");
            retrievedStudent.setAge(30);
            session.update(retrievedStudent);
            System.out.println("Student updated: " + retrievedStudent);
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

```
            // Delete
            session.delete(retrievedStudent);
            System.out.println("Student deleted.");

            // Commit transaction
            session.getTransaction().commit();
        } finally {
            // Close resources
            factory.close();
        }
    }
}
```

Output-

```
Student saved. ID: 1
Retrieved Student: Student{id=1, name='John Doe', age=25}
Student updated: Student{id=1, name='Updated Name', age=30}
Student deleted.
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

12. Perform JDBC crud operation

```
import java.sql.*;


public class JDBCCrudExample {

   // JDBC URL, username, and password of MySQL server

   private static final String JDBC_URL =
"jdbc:mysql://localhost:3306/your_database_name";

   private static final String USERNAME = "your_username";

   private static final String PASSWORD = "your_password";


   public static void main(String[] args) {

      try {

         // Register JDBC driver

         Class.forName("com.mysql.cj.jdbc.Driver");


         // Establish a connection

         Connection connection = DriverManager.getConnection(JDBC_URL, USERNAME,
PASSWORD);


         // Create a Statement

         Statement statement = connection.createStatement();


         // Create table if not exists

         statement.executeUpdate("CREATE TABLE IF NOT EXISTS students (id INT
PRIMARY KEY AUTO_INCREMENT, name VARCHAR(255), age INT)");


         // CRUD operations


         // Create

         statement.executeUpdate("INSERT INTO students (name, age) VALUES ('John Doe',
25)", Statement.RETURN_GENERATED_KEYS);

         ResultSet generatedKeys = statement.getGeneratedKeys();
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

```
            int newId = 0;
            if (generatedKeys.next()) {
                newId = generatedKeys.getInt(1);
                System.out.println("Student created. ID: " + newId);
            }

            // Read
            ResultSet resultSet = statement.executeQuery("SELECT * FROM students WHERE
id=" + newId);
            if (resultSet.next()) {
                System.out.println("Retrieved Student: " +
                        "ID=" + resultSet.getInt("id") +
                        ", Name=" + resultSet.getString("name") +
                        ", Age=" + resultSet.getInt("age"));
            }

            // Update
            statement.executeUpdate("UPDATE students SET age=30 WHERE id=" + newId);
            System.out.println("Student updated.");

            // Delete
            statement.executeUpdate("DELETE FROM students WHERE id=" + newId);
            System.out.println("Student deleted.");

            // Close resources
            statement.close();
            connection.close();
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
    }
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

```
}
```

Output-

```
Student created. ID: 1
Retrieved Student: ID=1, Name=John Doe, Age=25
Student updated.
Student deleted.
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

13.Perform Spring crud operation.

```
// Import the necessary packages
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.web.bind.annotation.*;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

// Define the entity class for the database table
@Entity
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String email;

    // Constructors, getters, and setters
    public User() {
    }

    public User(String name, String email) {
        this.name = name;
        this.email = email;
    }
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

```
    public Long getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public String getEmail() {
        return email;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}

// Define the repository interface for CRUD operations
public interface UserRepository extends JpaRepository<User, Long> {
}
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

```
// Define the controller class for RESTful API
@RestController
@RequestMapping("/users")
public class UserController {
    @Autowired
    private UserRepository userRepository;

    // Create a new user with POST request
    @PostMapping
    public User createUser(@RequestBody User user) {
        return userRepository.save(user);
    }

    // Read all users with GET request
    @GetMapping
    public Iterable<User> readAllUsers() {
        return userRepository.findAll();
    }

    // Read a single user by id with GET request
    @GetMapping("/{id}")
    public User readUserById(@PathVariable Long id) {
        return userRepository.findById(id).orElse(null);
    }

    // Update a user by id with PUT request
    @PutMapping("/{id}")
    public User updateUserById(@PathVariable Long id, @RequestBody User user) {
        User existingUser = userRepository.findById(id).orElse(null);
        existingUser.setName(user.getName());
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

```
        existingUser.setEmail(user.getEmail());
        return userRepository.save(existingUser);
    }

    // Delete a user by id with DELETE request
    @DeleteMapping("/{id}")
    public void deleteUserById(@PathVariable Long id) {
        userRepository.deleteById(id);
    }
}

// Run the application
@SpringBootApplication
public class SpringBootCrudOperationApplication {
    public static void main(String[] args) {
        SpringApplication.run(SpringBootCrudOperationApplication.class, args);
    }
}
```

For the outputs-

```
curl -X POST -H "Content-Type: application/json" -d
'{"name":"Alice","email":"alice@example.com"}' http://localhost:8080/users

# Read all users
curl -X GET http://localhost:8080/users

# Read a single user by id (replace 1 with the actual id)
curl -X GET http://localhost:8080/users/1

# Update a user by id (replace 1 with the actual id)
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

```
curl -X PUT -H "Content-Type: application/json" -d
'{"name":"Bob","email":"bob@example.com"}' http://localhost:8080/users/1


# Delete a user by id (replace 1 with the actual id)
curl -X DELETE http://localhost:8080/users/1
```

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

14.Write a Java program to create and start multiple threads that increment a shared counter variable concurrently.

```
public class Counter
 {
  private int count = 0;
  public synchronized void increment()
{
   count++;
  }
  public int getCount()
{
   return count;
  }
          }
public class Main
{
  public static void main(String[] args)
{
   Counter counter = new Counter();
   int numThreads = 6;
   int incrementsPerThread = 10000;
   IncrementThread[] threads = new IncrementThread[numThreads];

   // Create and start the threads
   for (int i = 0; i < numThreads; i++)
{
    threads[i] = new IncrementThread(counter, incrementsPerThread);
    threads[i].start();
   }

   // Wait for all threads to finish
   try
{
    for (IncrementThread thread: threads)
{
      thread.join();
    }
   }
catch (InterruptedException e)
{
    e.printStackTrace();
   }
   // Print the final count
   System.out.println("Final count: " + counter.getCount());
  }
}
```

Output- Final count: 60000

Abrar Rafik Mirje
Roll No: 58
PRN - 2023012101060

15.Write a Java program to create a base class Animal (Animal Family) with a method called Sound(). Create two subclasses Bird and Cat. Override the Sound() method in each subclass to make a specific sound for each animal.

```
public class Animal {
    public void makeSound() {
        System.out.println("The animal makes a sound");
    }
        }

public class Bird extends Animal {
    @Override
    public void makeSound() {
        System.out.println("The bird chirps");
    }
        }

public class Cat extends Animal {

    @Override

    public void makeSound() {

        System.out.println("The cat meows");

    }

}

public class Main {
    public static void main(String[] args) {
        Animal animal = new Animal();
        Bird bird = new Bird();
        Cat cat = new Cat();

        animal.makeSound(); // Output: The animal makes a sound
        bird.makeSound();   // Output: The bird chirps
        cat.makeSound();    // Output: The cat meows
    }
}
```

Output-

The animal makes a sound

The bird chirps

The cat meows