Name-Abhi Mehta
Reg No.-221080001
Branch-IT

Laboratory-01

Experiment-Write a Java program for the shown class diagram.

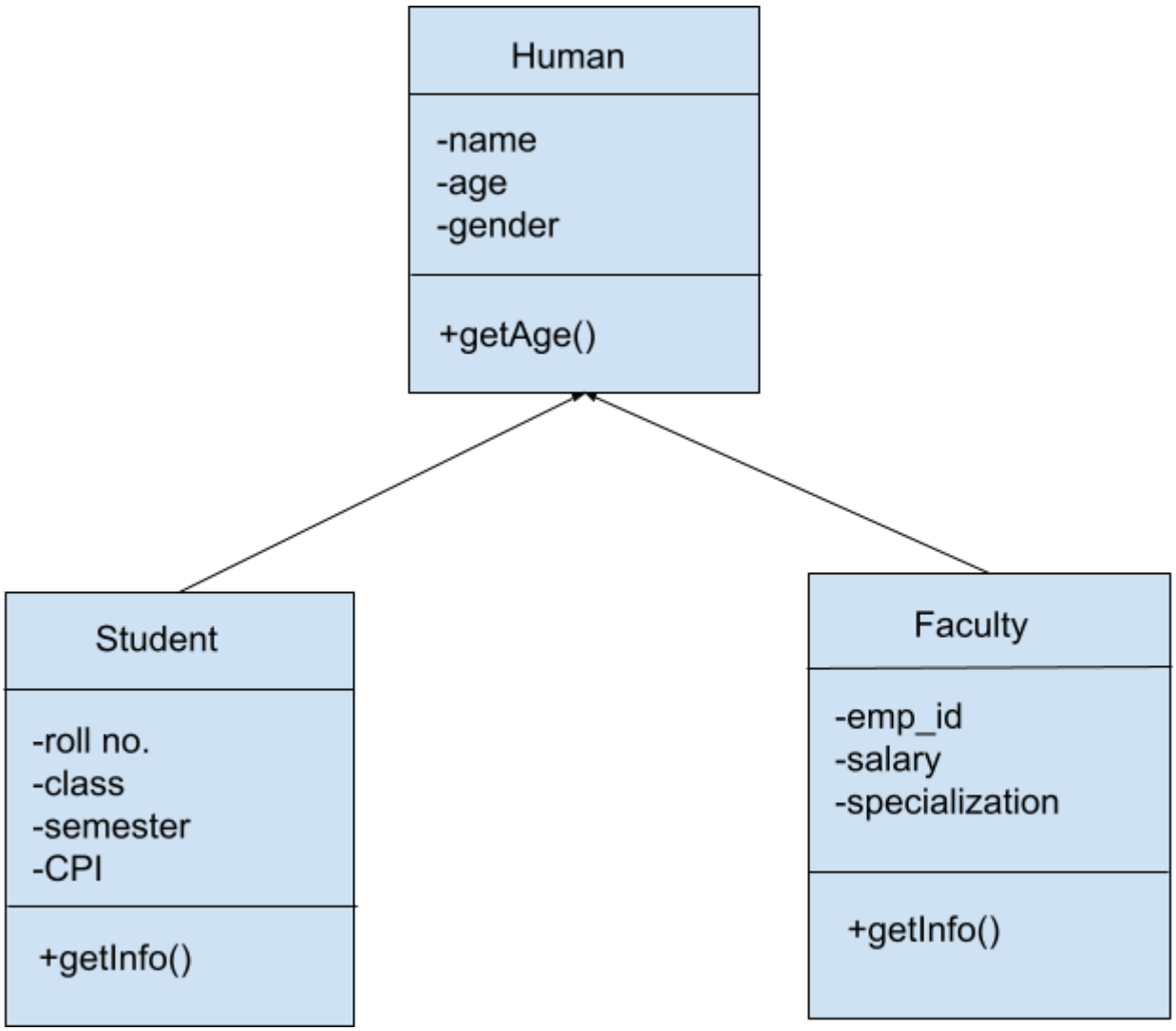
Different Constructors used in the experiment :

Human Constructor:
○ The Human class constructor takes three parameters: name, age, and gender.
○ It initializes the private member variables name, age, and gender with the values provided as arguments.
○ This constructor is used to create objects of the Human class with the specified attributes.

Student Constructor:
○ The Student class constructor extends the Human class and takes seven parameters: name, age, gender, rollno., studentClass, semester, and cpi.
○ It first calls the constructor of the superclass (Human) using super(name, age, gender) to initialize the common attributes.
○ Then, it initializes the additional attributes specific to the Student class: rollno., studentClass, semester, and cpi.
○ This constructor is used to create objects of the Student class with both general human attributes and student-specific attributes.

Faculty Constructor:
○ The Faculty class constructor extends the Human class and takes six parameters: name, age, gender, emp_id, salary, and specialization.
○ It first calls the constructor of the superclass (Human) using super(name, age, gender) to initialize the common attributes.
○ Then, it initializes the additional attributes specific to the Faculty class: emp_id, salary, and specialization.○ This constructor is used to create objects of the Faculty class with both general human attributes and faculty-specific attributes.

Class Diagram:



Code :

```
import java.util.Scanner;
class Human {
private String name;
private String age;
private String gender;
public Human(String name, String age, String gender) {
this.name = name;
this.age = age;
this.gender = gender;
}
public String getAge() {
return age;
}
public void setAge(String age) {
this.age = age;
}
public String getName() {
return name;
}
public void setName(String name) {
this.name = name;
}
public String getGender() {
return gender;
}
public void setGender(String gender) {
this.gender = gender;
}
public String getinfo() {
return "Name: " + getName() + "\nAge: " + getAge() + "\nGender: " +
getGender();
}
```

```
}
class Student extends Human {private String rollno;
private String studentClass;
private String semester;
private String cpi;
public Student(String name, String age, String gender, String rollno,
String studentClass, String semester, String cpi) {
super(name, age, gender);
this.rollno = rollno;
this.studentClass = studentClass;
this.semester = semester;
this.cpi = cpi;
}
public void updateInfo(String name, String age, String gender, String
rollno, String studentClass, String semester, String cpi) {
setName(name);
setAge(age);
setGender(gender);
this.rollno = rollno;
this.studentClass = studentClass;
this.semester = semester;
this.cpi = cpi;
}
@Override
public String getinfo() {
return super.getinfo() + "\nRoll No: " + rollno + "\nClass: " +
studentClass + "\nSemester: " + semester + "\nCPI: " + cpi;
}
}
class Faculty extends Human {
private String emp_id;
private String salary;
private String specialization;
public Faculty(String name, String age, String gender, String emp_id,
String salary, String specialization) {
super(name, age, gender);
this.emp_id = emp_id;
this.salary = salary;
this.specialization = specialization;}
@Override
public String getinfo() {
return super.getinfo() + "\nEmployee ID: " + emp_id + "\nSalary: " +
salary + "\nSpecialization: " + specialization;
}
```

```
}
public class Main {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
Faculty f = new Faculty("Abhi", "29", "male", "23", "1000000",
"IoT");
System.out.println(f.getinfo());
System.out.println();
Student s = new Student("Aman", "23", "male", "49", "IT", "4",
"8.38");
System.out.println(s.getinfo());
System.out.println();
System.out.println("Enter name: ");
String updatedName = sc.nextLine();
System.out.println("Enter age: ");
String updatedAge = sc.nextLine();
System.out.println("Enter gender: ");
String updatedGender = sc.nextLine();
System.out.println("Enter roll no: ");
String updatedRollno = sc.nextLine();
System.out.println("Enter class: ");
String updatedClass = sc.nextLine();
System.out.println("Enter semester: ");
String updatedSemester = sc.nextLine();
System.out.println("Enter CPI: ");
String updatedCpi = sc.nextLine();
s.updateInfo(updatedName, updatedAge, updatedGender, updatedRollno,
updatedClass, updatedSemester, updatedCpi);System.out.println("\nUpdated Student
Information:");
System.out.println(s.getinfo());
}
}
```

Learnings :

Classes and Objects:

○ Human, Student, and Faculty are classes that define the structure and behavior of objects.

○ Objects of these classes are created using constructors, e.g., new Faculty(...), new Student(...), and new Human(...).

Inheritance:

○ The Student and Faculty classes inherit properties and methods from the Human class using the extends keyword.

○ Inheritance allows code reuse and establishes an "is-a" relationship between subclasses and the superclass.

Constructor:

○ Constructors (Human, Student, Faculty) are special methods that initialize object properties when an object is created.

○ Constructors have the same name as the class and are used to set initial values.

Methods:

○ Methods are defined within classes and encapsulate behavior.

○ getinfo() methods in Human, Student, and Faculty classes return formatted information about objects.

Method Overriding:

○ In the Student and Faculty classes, getinfo() is overridden to provide customized information while retaining the same method signature as the superclass.

Encapsulation:

○ Data members (e.g., name, age, gender) are encapsulated as private fields within classes.

○ Accessor methods (e.g., getAge(), setAge()) allow controlled access to these fields.

Input/Output:

○ System.out.println() is used to display output to the console.

○ Input is obtained from the user using Scanner and displayed using System.out.println()

- We learnt the use of keyword super and well as the method to write the code related to the inheritance .

Errors :

1.Initially, I attempted to update student information within the getinfo function, which caused a mismatch in the number of arguments.To resolve this issue, introduced a new function called updateInfo specifically designed to handle user input for updating student details. This approach ensures that the getinfo function remains focused on retrieving and displaying information without any parameter mismatches.

2.While obtaining user input, I encountered an error related to type casting when trying to convert input directly into integers. I chose to initialize every variable as a string. This avoids the need for type casting since the input is already in string form, and I can then parse or convert it to the appropriate data types as needed.

3.I faced an error while attempting to directly access private members of a class using the class object.To solve this issue, I introduced getter functions to access the values of private members.

4.Some silly errors like indentation were encountered.

5. As java is case sensitive , some errors were encountered.

Conclusion :

In this lab experiment, we used the concept of multiple inheritance to write a code in java.I gained valuable knowledge about Java programming. I acquired a solid understanding of concepts such as classes,inheritance, and the implementation of methods. This experience allowed me to explore the use of superclasses, work with different data types, and handle type casting effectively. Overall, I was able to develop practical skills in the Java programming language.