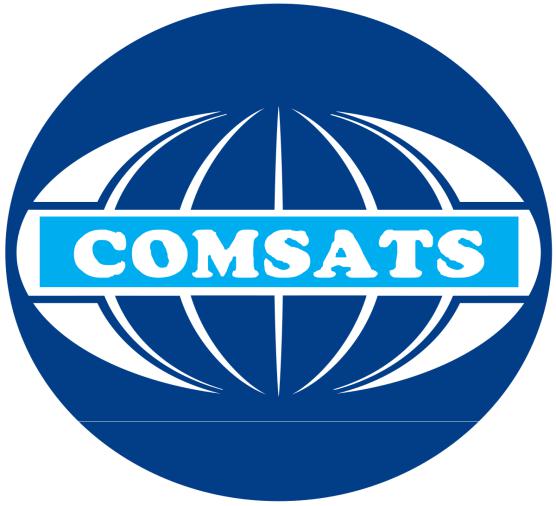
**Object Oriented Programming LAB**

****

**Assignment 2nd**

**Submitted to: Muhammad Ali-Khan**

**Submitted by: Nasratullah (FA18-BCS-401)**

**Submission Date: 26 April,2021.**

**Section: B**

**Q1**. A Computer Science department keeps track of its CS students using some custom software. Each student is represented by a Student object that features a pass() method that returns true if and only if the student has all 20 ticks to pass the year. The department suddenly starts teaching NS students, who only need 10 ticks to pass. Using inheritance and polymorphism, show how the software can continue to keep all Student objects in one list in code without having to change any classes other than Student.

**Employee Class:**

|  |
| --- |
| package practiceS2;  public abstract class Student {  private String name;  protected int ticksAquired = 0;  protected int ticksRequired = 0;  public boolean pass()  *// pass method is used to find if the number of tick acquired are equal to the number of tick required to pass the subjects.* {  return (ticksAquired >= ticksRequired);  }  protected abstract String getType();   public Student(String studentName) {  name = studentName;  }  public String toString() {  return name + ", type: " + getType() + ", ticks-required: " + ticksRequired + ", ticksAquired: " + ticksAquired + ", Result: " + pass();  } } |

**CS\_Student Class:**

|  |
| --- |
| package practiceS2;  public class CS\_Student extends Student {  *// This is a parametrized constructor* public CS\_Student(String studentName, int alreadyHave) {  super(studentName);  ticksRequired = 20;  ticksAquired = alreadyHave;  }  @Override *// override the function* protected String getType() {  return "CS";  }   } |

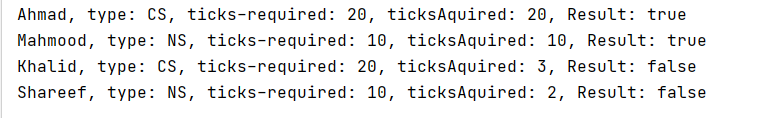
**NS\_Student Class**:

|  |
| --- |
| package practiceS2;  public class NS\_Student extends Student {  *// This is a parametrized constructor* public NS\_Student(String studentName, int alreadyHave) {  super(studentName);  ticksRequired = 10;  ticksAquired = alreadyHave;  }  @Override *// override the function* protected String getType() {  return "NS";  }  } |

**Test Class:**

|  |
| --- |
| package practiceS2;   import java.util.ArrayList; import java.util.List;  public class Main {  public static void main(String[] args) {  List<Student> studentList = new ArrayList<>();   CS\_Student CSs = new CS\_Student("Ahmad", 20);  studentList.add(CSs);   NS\_Student NSs = new NS\_Student("Mahmood", 10);  studentList.add(NSs);   *// Example data* CSs = new CS\_Student("Khalid", 3);  studentList.add(CSs);   NSs = new NS\_Student("Shareef", 2);  studentList.add(NSs);   *// Output* studentList.forEach(System.*out*::println);    }   } |

**Output:**

****

Q2. Imagine you have two classes: Employee (which represents being an employee) and Ninja (which represents being a Ninja). An Employee has both state and behaviour; a Ninja has only behaviour. You need to represent an employee who is also a ninja (a common problem in the real world). By creating only one interface and only one class (NinjaEmployee), show how you can do this without having to copy method implementation code from either of the original classes.

**Program:**

**Employee Class:**

|  |
| --- |
| package practiceS2;  public class Employee implements Check {  public void Behaviour() {   System.*out*.println("This is Employee behaviour");   } } |

**Ninja Class:**

|  |
| --- |
| package practiceS2;  public class Ninja extends Employee {  } |

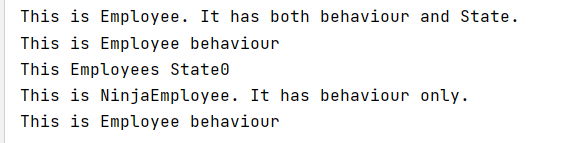
**Check Interface:**

|  |
| --- |
| package practiceS2;  public interface Check {  int *State*=0;   public void Behaviour(); } |

**Test Class:**

|  |
| --- |
| package practiceS2;  public class Main {   public static void main(String[] args) {  Employee ob=new Employee();   System.*out*.println("This is Employee. It has both behaviour and State.");   ob.Behaviour();   System.*out*.println("This Employees State"+ob.*State*);   Ninja ob1=new Ninja();   System.*out*.println("This is NinjaEmployee. It has behaviour only.");   ob1.Behaviour();   }  } |

**Output:**

****