CS6308 –JAVA PROGRAMMING LAB
INHERITANCE AND INTERFACES
DATE: 16.3.23

<u>Question A</u>

Create an abstract class called Employee with members:Empid(int), name(string), designation(string), DateofJoining(Date); constructor with arguments, toString(), with abstract method double calculateSal(). (2 marks)

```
interface CalBonus{
double calBonus();
}
```
(1 mark)

Derive a class called PermanentEmployee from Employee with fields: basicPay(double), gradePay(double), bonus(double), methods: constructor, toString(), override double calculateSalary that returns the salary by adding basicPay(double), gradePay(double)and bonus (3 marks)

PermanentEmployee implements interface CalBonus that overrides calBouns() by using the following calculations: (2 marks)

if basicPay greater than 10,000, bouns 5% of salary
if basicPay less than 10,000, bouns 10% of salary
if basicPay less than 5,000, bouns 15% of salary.

Derive a class called Salesperson from Employee whose data members are: basicSalary(double), SalesAmount(double) and noofSales(integer). The member functions are:

a. constructor with three arguments to initialise member variables age, name and salary; SalesAmount and noofSales is initialised to zero
b. addSale(double amount) – method to increment the number of sales and add the amount made to the SalesAmount
c. int getSales() – to return the sales made by the Salesperson
d. double getTotalSalesAmount()- to return the TotalSalesAmount made by the Salesperson
e. toString – to display the details of the Salesperson (3 marks)
f. implement interface CalBonus that overrides calBouns() to return the bonus by using the following calculations: (2)

SalesAmount	Bonus amount
10000-20000	1000
20000 – 40000	2000
>40000	5000

g. override calculateSal() to return the salary by adding bonus with basicSalary. (1)

Define **TestEmployee** that instantiates objects of Manager and SalesPerson**.** The methods of the derived class are tested. Check the SalesPerson object is an instance of class **Student. (2)**

<u>Question B</u>

Create an abstract class called **Account** that has datamembers: **accno**(int), **accname**(String) and **balance**(double), contactAddress(Address). The member methods are: parameterized constructor and **toString** method. abstract methods : **void deposit(double amt)** and **double withdraw(double amt)**
(3 marks)

Derive two classes from class **Account**: **SavingsAccount** and **CheckingsAccount**. The members of **SavingsAccount** are **noofTransactions**(int), parameterized constructor, overridden methods **toString()**, **deposit(double amt)** that increments balance with amt, **withdraw(double amt)** that decrements the balance by amt and **noofTransactions** is incremented by 1 inside **deposit()** and **withdraw()**.The members of **CheckingAccount** are parameterized constructor, overridden methods **deposit(double amt)** that increments balance with amt, **withdraw(double amt)** that decrements the balance by amt only if the balance is above 1000 after decrementation and **toString().** (5 marks)

Define **TestAccount** that instantiates objects of **SavingsAccount** and **CheckingsAccount.** The methods of the derived class are tested. Check the **SavingsAccount** object is an instance of class **Account.**
(2 marks)

```
public interface Calculate{
            public static final int totalTrans = 3;
            double interest_rate = 0.15;
            public double cal_interest();
    }
```
(1 marks)

Implement the following interface in SavingsAccount and CheckingAccount. The totalTrans gives the total number of transactions the SavingsAccount object can perform while the cal_interest is overridden in CheckingsAccount to calculate the interest for the balance (using the formula PNR/100). (4 marks)

<u>Question C</u>

Create an class called Course with fields: courseName(string), credits(int) - the value of credits can be 3 or 4, contactHrs(int), Grade(String), type(String) which indicates whether core or elective; Constructor with args, toString() and calcuateGrade(double) – which initialises the grade based on the mark passed as argument as given below :   (2 marks)

0-49 – U
50- 60 –D
61 -70 – C
71-80 – B
81-90 – A
91-100 - O
Create an abstract class called Student with fields: name(String), cgpa(double), rollNo(int); coursesEnrolled <Course> - arraylist of courses; Constructor with three arguments for initialising name, cgpa and rollNo inside which arraylist of courses created but null list and toString(),

abstract methods :void enrollCourse(Course), void dropCourse(Course), bool findCourse(Course),   (2 marks)

Derive a class called FullTimeStudent from Class Student with static member: maxCredits(int) initialized to 24 indicating the maximum number of credits the FullTimeStudent can enroll per semester; constructor with arguments, static method: void modifyCredits(int) that modifies the value of maxCredits.

Override the following methods: enrollCourse(Course) that adds courseEnrolled<Courses> such that the number of credits doesnot exceed maxCredits and also it should not been already enrolled; dropCourse(Course) that removes the Course passed as argument from courseEnrolled<Courses>; bool findCourse(Course) – returns true or false if the given course existes in the courseEnrolled<Courses>, toString- display the details of FullTimeStudent.(4 marks)

Derive a class called PartTimeStudent from Class Student with static member: maxCourses(int) initialied to 3 that indicates the maximum number of Courses that PartTimeStudent can enroll per semester; constructor with arguments, static method: void modifyNoCourses(int) that modifies the value of maxCourses

Override the following methods: enrollCourse(Course) that adds courseEnrolled<Courses> such that the number of courses doesnot exceed maxCredits and also it should not been already enrolled; dropCourse(Course) that removes the Course passed as argument from courseEnrolled<Courses>; bool findCourse(Course) – returns true or false if the given course exists in the courseEnrolled<Courses>, toString- display the details of PartTimeStudent. (4 marks)

Define **TestStudent** that instantiates objects of PartTimeStudent and FullTimeStudent**.** The methods of the derived class are tested. Check the PartTimeStudent object is an instance of class **Student.** **(2 marks)**

```
public interface Calculate{
        public static final double cal_CGPA();
    }
```
(1 marks)

Implement the following interface in FullTimeStudent and PartTimeStudent. The cal_CGPA()t is overridden in FullTimeStudent and PartTimeStudent to calculate cgpa of the respective objects. (2 marks)

Question D

Create an abstract class Player with fields: id(int) – indicates the id number of the Player, balance(double) – the amount of money, gamesPlayed(int)- indicates the number or games played by Player; Constructor with arguments in which the gamesPlayed is initialised to zero, toString(), abstract method: void incrgamesPlayed() and calBonus() (2)

Derive a class PremiumPlayer with fields: points(int) initialised to zero in constructor with arguments, toString()

abstract method: void incrgamesPlayed() to increment the value of gamesPlayed whenever a game is started to play by a value of one.

calBonus() – increments the balance by the value passed as argument (3)

```
interface MineSweeper{
public static final int gameDuration = 60;
public static final int wonpoints = 5;
bool startGame(gameDuration);
int statusGame(String);
}
```
(2)

```
interface Cal_Bonus{
public static final int bonusperPoint = 20;
int calBonus();
}
```
(2)

The class PremiumPlayer implements the following interfaces MineSweeper and Cal_Bonus. Override the methods such that:

bool startGame() starts incrementing the gameDuration from 0 until 60 and returns 1 if game finished void statusGame(String) – that increments the points by wonpoints if the argument passed is WIN or else LOST no change in points

int calBonus() – which returns the bonus obtained by the PremiumPlayer by mutlipyng the bonusperpoint and points.
(4)

Define TestPlayer that instantiates objects of PremiumPlayer**.** The methods of the derived class are tested. Check the PremiumPlayer object is an instance of class Player**.** **(2 marks)**