Object Oriented Programing

**Constructor Programs - I**

Creating objects of Employee

Create a class Employee (Business Logic Class)

Instance Variables:

employeeId :private-int

employeeName : private-String

employeeSalary: private-double

Create a parameterized constructor to initialilize all the fields.

Create a static method getEmployeeObject() in the Employee class

 Method Name : getEmployeeObject()

  Parameter : no parameter

 Return type : Employee

Access modifier : public

getEmployeeObject() method of Employee class should take employeeId, employeeName, employeeSalary from the keyboard to initialize the instance variables, Create and return the Employee object from this method.

Override toString() from Object class to print Employee class properties.

Create another class Main which contains main method (ELC class). Accept 5 Employee Object from keyboard and print their data using toString() method

**import** java.util.Scanner;

**class** Employee

{

**private** **int** employeeId ;

**private** String employeeName ;

**private** **double** employeeSalary;

Employee(**int** id,String name,**double** sal)

{

employeeId=id;

employeeName=name;

employeeSalary=sal;

}

**public** **static** Employee getEmployeeObject()// return type class

{

Scanner sc=**new** Scanner(System.***in***);

System.***out***.print("Enter employee Id :");//println method

**int** id=sc.nextInt();

sc.nextLine(); // Consume the newline character

System.***out***.print("Enter employee name :");

String name= sc.nextLine();//read the name

System.***out***.print("Enter employee salary :");

**double** sal=sc.nextDouble();

**return** **new** Employee(id,name,sal);//return Employee object

}

@Override

**public** String toString()

{

**return** "Employee [employeeId=" + employeeId + ", employeeName=" + employeeName + ", employeeSalary=" + employeeSalary + "]";

}

}

//shortcut

//number lock off and press zero to bold cursor off

//alt+sht + s //generate toString

//alt +shft +s //constructor

//this -current class type non static reference var ,this return non static

//hold current class object

//if we pass object in print it calls toString

**public** **class** MyProgram

{

**public** **static** **void** main(String[] args)

{

//accept employee objects from keyboard

Employee emp1 = Employee.*getEmployeeObject*();

System.***out***.println(emp1);

Employee emp2 = Employee.*getEmployeeObject*();

System.***out***.println(emp2);

Employee emp3 = Employee.*getEmployeeObject*();

System.***out***.println(emp3);

Employee emp4 = Employee.*getEmployeeObject*();

System.***out***.println(emp4);

Employee emp5 = Employee.*getEmployeeObject*();

System.***out***.println(emp5);

//////////////////////////////////////////////////////////

//alt+shft+/

/\*

\* Employee[] employees = new Employee[5]; // Array to store 5 employees

\*

\* // Accept 5 Employee Objects from keyboard

\* for (int i = 0; i < 5; i++)

\* {

\* System.out.println("Enter details for Employee " + (i + 1));

\* employees[i] = Employee.getEmployeeObject();

\* }

\*

\* // Print their data using toString() method

\* System.out.println("\nEmployee details:");

\* for (int i = 0; i < 5; i++)

\* {

\* System.out.println(employees[i].toString());

\* }

\*/

}

}

**Output :-**

Enter employee Id :101

Enter employee name :sourab das

Enter employee salary :140000

Employee [employeeId=101, employeeName=sourab das, employeeSalary=140000.0]

Enter employee Id :102

Enter employee name :bajrang singh

Enter employee salary :4500000

Employee [employeeId=102, employeeName=bajrang singh, employeeSalary=4500000.0]

Enter employee Id :103

Enter employee name :srinivas

Enter employee salary :5000

Employee [employeeId=103, employeeName=srinivas, employeeSalary=5000.0]

Enter employee Id :104

Enter employee name :bhawesh giri

Enter employee salary :96000

Employee [employeeId=104, employeeName=bhawesh giri, employeeSalary=96000.0]

Enter employee Id :105

Enter employee name :abhishek kumar

Enter employee salary :80000

Employee [employeeId=105, employeeName=abhishek kumar, employeeSalary=80000.0]

Using this keyword as argument

Create a BLC class called A

 Instance variable

 private int data = 15;

 Create a no-argument constructor, where create the object for class B (Another BLC class) and call the non-static display method available in class B.

 Create an instance method show() in class A, which will print instance variable data.

 Create an another BLC class B

 Instance variable

 private A obj;

 Create a parameterized constructor which takes class A as a parameter to initialize the instance variable

 Create a display method inside class B which internally calling the show() method of class A

 Create an ELC class Test which contains main method to test the application.

 Note :- BLC and ELC all the classes must be declared as public.

 Expected Output:

instance variable data in A = 15

//BLC and ELC all the classes must be declared as public.

**class** A

{

**private** **int** data = 15;

//shotcut

//alt+shift+s+o

**public** A()

{

B objb=**new** B(**this**); // if we pass new () we will get stack overflow

//B objb=new B(new A());

objb.display();

}

// Instance method show() to print instance variable data

**void** show()

{

System.***out***.println("Instance variable data in A ="+data);

}

}

**class** B

{

**private** A obj; //stores A class object

**public** B(A obj)// constructor takes class A as a parameter

{

**this**.obj=obj;// initialize instance variable

}

**void** display()

{

obj.show();//calling class A show method

//new A().show();//stack overflow

}

}

**class** Test

{

/\*

\* public static void main(String[] args)

\* {

\* A objA = new A();

\* }

\*/

}

**public** **class** MyProgram

{

**public** **static** **void** main(String[] args)

{

A objA = **new** A();

}

}

**Output :-**

Instance variable data in A =15

Constructor chaining

Your task is to design a Java class named Book that encapsulates information about books. The class should include attributes such as title, author, and publication year. Implement multiple constructors for enhanced flexibility, logical methods to determine if a book is considered a classic or to calculate the number of years since its publication, and detailed information about each attribute.

The Book class should include the following elements:

Attributes:

private title (String): The title of the book.

private author (String): The author of the book.

private publicationYear (int): The year the book was published.

Constructors:

A constructor that initializes the title only.

A constructor that initializes both the title and author.

A full constructor that initializes the title, author, and publication year.

Logical Methods:

A method named isClassic that determines if a book is considered a classic. This method should return a boolean (true if published before year 2000, false otherwise).

A method named calculateYearsSincePublication that takes the current year as a parameter and returns an integer representing the number of years since the book's publication.

Main class :

In the main method, demonstrate the usage of the Book class by creating instances with different constructors and utilizing the logical methods. Output relevant information for each book, including title, author, publication year, and the results of the logical methods.

Additional Information:

Note : If User is creating an object by calling One parameterized constructor then initialize the variables through this(), with given default value(author as Unknown and publicationYear as current year).

    If the user is creating an object by calling Two parameterized constructor then initilize the publicationYear through this() with default year(current year).

**import** java.util.Calendar;

**class** Book

{

**private** String title ;//(String): The title of the book.

**private** String author;// (String): The author of the book.

**private** **int** publicationYear; // (int): The year the book was published.

**public** Book(String title) //Alt+shift+s+o /move shortcut //Alt+up

{

**this**.title=title;

**this**.author = "Unknown";

**this**.publicationYear = Calendar.*getInstance*().get(Calendar.***YEAR***);

}

**public** Book(String title, String author)//Alt+Shift+S+O

{

**this**(title); //initialize the variables through this(),

**this**.author = author;

}

**public** Book(String title, String author, **int** publicationYear)

{

**this**.title = title;

**this**.author = author;

**this**.publicationYear = publicationYear;

}

**boolean** isClassic()

{

**if**(publicationYear<2000)**return** **true**;

**else** **return** **false**;

//return publicationYear < 2000;

}

**int** calculateYearsSincePublication(**int** currentYear)

{

**return** currentYear-publicationYear;

}

//Getters //alt+shift+s

**public** String getTitle()

{

**return** title;

}

**public** String getAuthor()

{

**return** author;

}

**public** **int** getPublicationYear()

{

**return** publicationYear;

}

}

**public** **class** MyProgram

{

**public** **static** **void** main(String[] args)

{

// Demonstration

**int** currentYear = Calendar.*getInstance*().get(Calendar.***YEAR***);

// Create instances using different constructors

Book book1 = **new** Book("Title Only");

Book book2 = **new** Book("Title and Author", "Hardin");

Book book3 = **new** Book("Full Book", "Tessa", 1995);

// Output relevant information

System.***out***.println("Book 1:");

System.***out***.println("Title: " + book1.getTitle());

System.***out***.println("Author: " + book1.getAuthor());

System.***out***.println("Publication Year: " + book1.getPublicationYear());

System.***out***.println("Is Classic? " + book1.isClassic());

System.***out***.println("Years Since Publication: " + book1.calculateYearsSincePublication(currentYear));

System.***out***.println();

System.***out***.println("Book 2:");

System.***out***.println("Title: " + book2.getTitle());

System.***out***.println("Author: " + book2.getAuthor());

System.***out***.println("Publication Year: " + book2.getPublicationYear());

System.***out***.println("Is Classic? " + book2.isClassic());

System.***out***.println("Years Since Publication: " + book2.calculateYearsSincePublication(currentYear));

System.***out***.println();

System.***out***.println("Book 3:");

System.***out***.println("Title: " + book3.getTitle());

System.***out***.println("Author: " + book3.getAuthor());

System.***out***.println("Publication Year: " + book3.getPublicationYear());

System.***out***.println("Is Classic? " + book3.isClassic());

System.***out***.println("Years Since Publication: " + book3.calculateYearsSincePublication(currentYear));

}

}

**Output :-**

Book 1:

Title: Title Only

Author: Unknown

Publication Year: 2024

Is Classic? false

Years Since Publication: 0

Book 2:

Title: Title and Author

Author: Hardin

Publication Year: 2024

Is Classic? false

Years Since Publication: 0

Book 3:

Title: Full Book

Author: Tessa

Publication Year: 1995

Is Classic? true

Years Since Publication: 29

Copying one object data in another Object

Programming Question: Student Class with Grade Calculation

Your task is to design a Java class named Student that encapsulates information about students, including name, ID, course, fee, grade, and average. Your class should include a constructor for creating a student object by copying another student object and a method to calculate the grade based on the average.

The Student class should have the following elements:

Constructors:

A parameterized constructor that initializes a student object by copying another student object.

A parameterized constructor that initializes a student object with basic information, such as name, ID, course, and fee.

Method:

A method named calculateGrade that initialize the grade of a student based on the average. Use the following grading scale:

A: Average >= 90

B: Average >= 80

C: Average >= 70

D: Average >= 60

F: Average < 60

Attributes:

Name: String

ID: int

Course: String

Fee: double

Grade: char

Average: double

Main Class:

In the main method, demonstrate the usage of the Student class by creating instances with different constructors, setting average values, and calculating grades.

**class** Student

{

**private** String name;

**private** **int** id;

**private** String course;

**private** **double** fee;

**private** **char** grade;

**private** **double** average;

**public** Student(String name, **int** id, String course, **double** fee)//Alt+shift+s+o

{

**this**.name = name;

**this**.id = id;

**this**.course = course;

**this**.fee = fee;

}

**public** Student(Student otherStudent) //copying object

{

**this**.name = otherStudent.name;

**this**.id = otherStudent.id;

**this**.course = otherStudent.course;

**this**.fee = otherStudent.fee;

**this**.grade = otherStudent.grade;

**this**.average = otherStudent.average;

}

**void** calculateGrade()

{

**if**(average >= 90)grade='A';

**else** **if**(average >= 80)grade='B';

**else** **if**(average >= 70)grade='C';

**else** **if**(average >= 60)grade='D';

**else** **if**(average <60)grade='F';

}

// Getter and Setter methods

**public** String getName()

{

**return** name;

}

**public** **int** getId()

{

**return** id;

}

**public** String getCourse()

{

**return** course;

}

**public** **double** getFee()

{

**return** fee;

}

**public** **char** getGrade()

{

**return** grade;

}

**public** **double** getAverage()

{

**return** average;

}

**public** **void** setAverage(**double** average)

{

**this**.average = average;

}

}

**public** **class** MyProgram

{

**public** **static** **void** main(String[] args)

{

// Demonstration

// Create instances using different constructors

Student student1 = **new** Student("Bhawesh giri", 12345, "Computer Science", 1500.00);

Student student2 = **new** Student(student1); // Copy student1

// Set average values

student1.setAverage(85.5);

student2.setAverage(72.0);

// Calculate grades

student1.calculateGrade();

student2.calculateGrade();

// Output relevant information

System.***out***.println("Student 1:");

System.***out***.println("Name: " + student1.getName());

System.***out***.println("ID: " + student1.getId());

System.***out***.println("Course: " + student1.getCourse());

System.***out***.println("Fee: $" + student1.getFee());

System.***out***.println("Average: " + student1.getAverage());

System.***out***.println("Grade: " + student1.getGrade());

System.***out***.println();

System.***out***.println("Student 2:");

System.***out***.println("Name: " + student2.getName());

System.***out***.println("ID: " + student2.getId());

System.***out***.println("Course: " + student2.getCourse());

System.***out***.println("Fee: $" + student2.getFee());

System.***out***.println("Average: " + student2.getAverage());

System.***out***.println("Grade: " + student2.getGrade());

}

}

**Output :-**

Student 1:

Name: Bhawesh giri

ID: 12345

Course: Computer Science

Fee: $1500.0

Average: 85.5

Grade: B

Student 2:

Name: Bhawesh giri

ID: 12345

Course: Computer Science

Fee: $1500.0

Average: 72.0

Grade: C

Using Copy Constructor

Create a Java (BLC)class named Employee with following attributes:

private String name

private int age

private double salary and

private String department

Implement a parameterised constructor to initialize all the instance variable of Employee.

Implement a copy constructor for the Employee class that takes a Employee object and initialize the new Employee object with the same attributes.

Take a pair of setter and getter method for all the instance variables.

In the Employee class, create a method named raiseSalary(double percentage) that increases the salary of the employee by the specified percentage.

Create a ELC class and create Employee object and sets its attributes.

Then, use the raiseSalary method to increase the employee's salary by 10%. Create a copy of the Employee object using the copy constructor and print the attributes of both the original and the copy of the Employee objects to confirm that the copy constructor works correctly and that the raiseSalary method did not affect the attributes of the copy of the Employee object.

**class** Employee

{

**private** String name ;

**private** **int** age ;

**private** **double** salary ;

**private** String department;

**public** Employee(String name, **int** age, **double** salary, String department)

{

//constructor is used to initialize all the instance variable

**this**.name = name;

**this**.age = age;

**this**.salary = salary;

**this**.department = department;

}

// Copy Constructor

**public** Employee(Employee otherEmployee) //takes employee object

{

**this**.name = otherEmployee.name;

//initialize new employee object with same attributes

**this**.age = otherEmployee.age;

**this**.salary = otherEmployee.salary;

**this**.department = otherEmployee.department;

}

// Getter and Setter methods

**public** String getName()

{

**return** name;

}

**public** **void** setName(String name)

{

**this**.name = name;

}

**public** **int** getAge()

{

**return** age;

}

**public** **void** setAge(**int** age)

{

**this**.age = age;

}

**public** **double** getSalary()

{

**return** salary;

}

**public** **void** setSalary(**double** salary)

{

**this**.salary = salary;

}

**public** String getDepartment()

{

**return** department;

}

**public** **void** setDepartment(String department)

{

**this**.department = department;

}

// Method to raise salary by percentage

**public** **void** raiseSalary(**double** percentage)

{

**this**.salary \*= (1 + percentage / 100);

}

}

**public** **class** MyProgram

{

**public** **static** **void** main(String[] args)

{

// Create an Employee object and set its attributes

Employee originalEmployee = **new** Employee("Rahul Kumar", 30,50000.0, "Engineering");

// Use raiseSalary method to increase the employee's salary by 10%

originalEmployee.raiseSalary(10);

// Create a copy of the Employee object using the copy constructor

Employee copiedEmployee = **new** Employee(originalEmployee);

// Print the attributes of both original and copied Employee objects

System.***out***.println("Original Employee:");

System.***out***.println("Name: " + originalEmployee.getName());

System.***out***.println("Age: " + originalEmployee.getAge());

System.***out***.println("Salary: " + originalEmployee.getSalary());

System.***out***.println("Department: " + originalEmployee.getDepartment());

System.***out***.println();

System.***out***.println("Copied Employee:");

System.***out***.println("Name: " + copiedEmployee.getName());

System.***out***.println("Age: " + copiedEmployee.getAge());

System.***out***.println("Salary: " + copiedEmployee.getSalary());

System.***out***.println("Department: " + copiedEmployee.getDepartment());

}

}

**Output :-**

Original Employee:

Name: Rahul Kumar

Age: 30

Salary: 55000.00000000001

Department: Engineering

Copied Employee:

Name: Rahul Kumar

Age: 30

Salary: 55000.00000000001

Department: Engineering