## Introduction to OOPs in Java

* 1. Create a class Person with properties (name and age) with following features.

a. Default age of person should be 18;

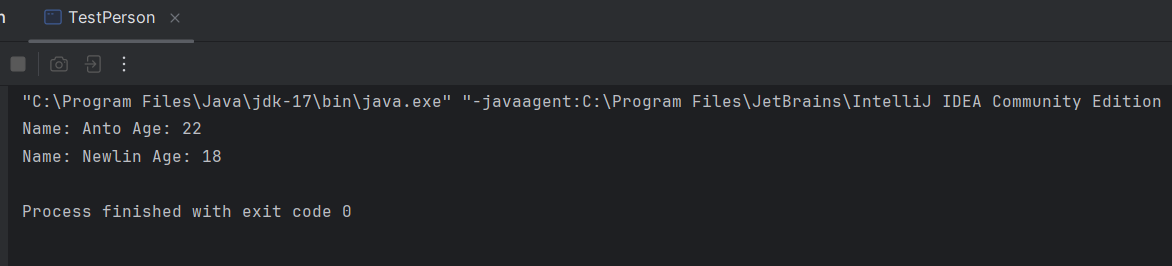
b. A person object can be initialized with name and age;

c. Method to display name and age of person

public class TestPerson { // class that contains main method  
 //Main method  
 public static void main(String[] args) {  
  
 // Creating an object of the Person class using parameterized constructor with two arguments.  
 Person anto=new Person("Anto",22);  
 // Creating an object of the Person class using parameterized constructor with one argument.  
 Person newlin=new Person("Newlin");  
 anto.display(); // calling display method for object 'anto'  
 newlin.display(); // calling display method for object 'newlin'  
 }  
}

public class Person {  
 // private instance variables for storing name and age  
 private String name;  
 private int age=18; // default age is set to 18  
  
 // Default constructor  
 public Person(){  
  
 }  
 // Constructor with one parameter  
 public Person(String name){  
 this.name=name; // assigns provided 'age' to the instance variable 'age'  
  
 }  
 // Constructor with two parameters  
 public Person(String name,int age){  
 this.name=name; // assigns provided 'name' to the instance variable 'name'  
 this.age=age; // assigns provided 'age' to the instance variable 'age'  
 }  
 // Method to display name and age  
 public void display(){  
 System.*out*.println("Name: "+name+" Age: "+age);  
 }  
}

Output:



1.2). Create class Product (pid, price, quantity) with parameterized constructor. Create a main function in different class (say XYZ) and perform following task:

a. Accept five product information from user and store in an array

b. Find Pid of the product with the highest price.

c. Create method (with array of product's object as argument) in XYZ class to calculate and return the total amount spent on all products. (amount spent on single product-price of product quantity of product)

public class Product {  
 // public instance variable for storing pid, price and quantity  
 public int pid; // Product ID  
 public int price;  
 public int quantity;  
 // Default constructor  
 public Product() {  
   
 }

// Constructor with three parameters  
 public Product(int pid,int price,int quantity) {  
 this.pid=pid; // assigns provided 'pid' to the instance variable 'pid'  
 this.price=price; // assigns provided 'price' to the instance variable 'price'  
 this.quantity=quantity; // assigns provided 'quantity' to the instance variable 'quantity'  
 }  
}

import java.util.Scanner;  
  
public class TestProduct { // class with main method  
 // Main method  
 public static void main(String[] args) {  
 Scanner obj=new Scanner(System.*in*); // Scanner class object to get user input  
  
 Product[] products =new Product[5]; // creates an array to store 5 Product objects  
 // Loop to read input for 5 products and create Product objects  
 for(int i=0;i<5;i++){  
 int pid=obj.nextInt(); // reads pid from user  
 int price=obj.nextInt(); // reads price user  
 int quantity=obj.nextInt(); // reads quantity user  
 products[i]=new Product(pid,price,quantity); // creating new Product object and store it in the array  
 }  
 // Initializing variables to find the product with the highest price  
 int highestPrice=Integer.*MIN\_VALUE*; // declare variable with small possible integer  
 int highestPriceIndex=-1; // highest price product as -1  
 //Loop to find the index of the highest price product  
 for (int i = 0; i < 5; i++) {  
 if (highestPrice < products[i].price) { // checks if the current product has higher price  
 highestPrice = products[i].price; // updates highest product price  
 highestPriceIndex = i; // updates index of highest product price  
 }  
 }  
 int costliestProduct=products[highestPriceIndex].pid; // gets the product id of product with the highest price  
 System.*out*.println("Product with highest price: "+costliestProduct); // displays pid of product with the highest price  
 System.*out*.println("Total amount spent: "+*totalAmount*(products)); // display total amount spent by calling total amount() method  
  
 }  
 // Method to calculate total amount spent on all products  
 public static int totalAmount(Product[] products){  
  
 int totalPrice=0; // Initialize total price to 0.  
 // Loops through product array and calculates total price  
 for(int i=0;i<5;i++){  
 totalPrice += products[i].price\*products[i].quantity; // add totalPrice with price \* quantity for each product  
 }  
 return totalPrice; // returns total amount spent on all products  
 }  
}

Output:

A screen shot of a computer

Description automatically generated

1.3) Create Class Account with data member As Balance. Create two constructors (no argument, and two arguments) and perform following task

a. method to deposit the amount to the account.

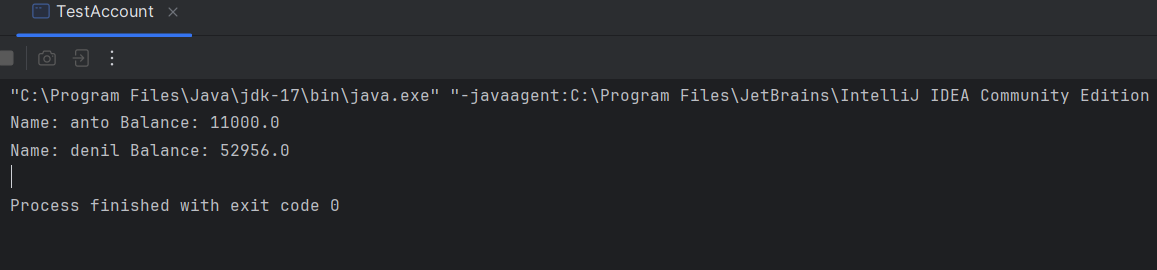
b. method to withdraw the amount from the account.

c. method to display the Balance

public class Account {  
 // Data members to store the accountHolder's name and balance  
 private String accountHolder;  
 private double balance;  
 // Default constructor  
 public Account(){  
  
 }  
 // Parameterized constructor with two parameters  
 public Account(String accountHolder,double initialBalance){  
 this.balance=initialBalance;  
 this.accountHolder=accountHolder;  
 }  
 // Method to deposit amount  
 public void deposit(double amount) {  
 balance+=amount;  
 }  
 // Method to withdraw amount  
 public void withdraw(double amount){  
 if(amount>0 && amount<=balance){ // checks whether amount in account is greater than or equal to withdrawal amount  
 balance-=amount;  
 }else{  
 System.*out*.println("Insufficient balance");  
 }  
 }  
 // Method to display balance  
 public String getBalance(){  
 return "Name: "+accountHolder +" Balance: " + balance;  
 }  
}

public class TestAccount { // class with main method  
 // Main method  
 public static void main(String[] args) {  
 // Creating an Account object acc1 with accountHolder's name and initial balance of 30000  
 Account acc1=new Account("anto",30000);  
 acc1.deposit(1000); // deposits 1000  
 acc1.withdraw(20000); // withdraws 20000  
 System.*out*.println(acc1.getBalance()); // displays balance  
 // Creating an Account object acc2 with accountHolder's name and initial balance of 30000  
 Account acc2=new Account("denil",50000);  
 acc2.deposit(3456); // deposits 3456  
 acc2.withdraw(500); // withdraws 500  
 System.*out*.println(acc2.getBalance()); //displays balance  
 }  
}

Output:



1.4) Define a base class Person with attributes name and age.

Create a subclass Employee that inherits from Person and adds attributes like employeeID and salary.

Use the super keyword to initialize the Person attributes in the Employee constructor

public class Employee extends Person {  
 public int employeeID; // Employee ID as public  
 private int salary; // Salary of the employee as private  
 // Default constructor  
 public Employee(){  
  
 }  
 //Parameterized constructor  
 public Employee(String name,int age,int employeeID,int salary){  
 super(name,age); // calls the constructor of the superclass to initialize name and age  
 this.employeeID=employeeID; // assigns the provided employeeID to the 'employeeID' instance variable  
 this.salary=salary; // assigns the provided salary to the 'salary' instance variable  
 }  
 // Getter method to access private salary variable  
 public int getSalary(){  
 return salary;  
 }  
 // Setter method to update the private salary variable  
 public void setSalary(int salary){  
 this.salary=salary;  
 }  
}

public class Person {  
 public String name; // name of the person  
 public int age; // age of the person  
 // Default constructor  
 public Person(){  
 }  
 // Parameterized constructor to initialize name and age  
 public Person(String name,int age){  
 this.name=name; // assigns provided 'name' to the 'name' instance variable  
 this.age=age; // assigns provided 'age' to the 'age' instance variable  
 }  
}

public class Test { // class with main method  
 // Main method  
 public static void main(String[] args) {  
 // Creating the Employee object with name, age, employee ID, and salary.  
 Employee employee1=new Employee("Anto",22,10001,50000); // creates first Employee object  
 Employee employee2=new Employee("Newlin",18,10002,80000); // creates second Employee object  
 // Display details of employee1  
 System.*out*.println("EmployeeID: "+employee1.employeeID+" Name: "+employee1.name+" Age: "+employee1.age+" Salary: "+employee1.getSalary());  
 // Display details of employee2  
 System.*out*.println("EmployeeID: "+employee2.employeeID+" Name: "+employee2.name+" Age: "+employee2.age+" Salary: "+employee2.getSalary());  
 }  
}

Output:

A screen shot of a computer

Description automatically generated