**1.1.Solution**

package taskoops;  
  
 class Person {  
 int age=18;  
 String name;  
  
 public void displayDetails(){  
 System.*out*.println("Person age is"+" "+this.age);  
 System.*out*.println("Person name is"+" "+name);  
 }  
 public Person(int age,String name){  
 this.age=age;  
 this.name=name;  
  
 }  
 public static void main(String[] args) {  
 Person person=new Person(18,"Ramesh");  
 person.displayDetails();  
  
  
 }  
  
}

**output:**

Person age 18

Person name Ramesh

**1.2 Solution**

package taskoops;  
  
import java.util.Scanner;  
  
class Product {  
 // Data members  
 private int pid;  
 private double price;  
 private int quantity;  
  
 // Parameterized constructor  
 public Product(int pid, double price, int quantity) {  
 this.pid = pid;  
 this.price = price;  
 this.quantity = quantity;  
 }  
  
 // Getters  
 public int getPid() {  
 return pid;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public int getQuantity() {  
 return quantity;  
 }  
  
 public static class Xyz {  
 // Method to calculate total amount spent on all products  
 public static double calculateTotalAmount(Product[] products) {  
 double totalAmount = 0;  
 for (Product product : products) {  
 totalAmount += product.getPrice() \* product.getQuantity();  
 }  
 return totalAmount;  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 // Array to store 5 products  
 Product[] products = new Product[5];  
  
 // Accept 5 product details from the user  
 for (int i = 0; i < 5; i++) {  
 System.*out*.println("Enter details for product " + (i + 1) + " (pid, price, quantity):");  
 int pid = scanner.nextInt();  
 double price = scanner.nextDouble();  
 int quantity = scanner.nextInt();  
  
 // Create a product and add it to the array  
 products[i] = new Product(pid, price, quantity);  
 }  
  
 // Find the product ID (pid) with the highest price  
 int maxPricePid = products[0].getPid();  
 double maxPrice = products[0].getPrice();  
  
 for (Product product : products) {  
 if (product.getPrice() > maxPrice) {  
 maxPrice = product.getPrice();  
 maxPricePid = product.getPid();  
 }  
 }  
  
 System.*out*.println("Product ID with the highest price: " + maxPricePid);  
  
 // Calculate and display the total amount spent  
 double totalAmount = *calculateTotalAmount*(products);  
 System.*out*.println("Total amount spent on all products: " + totalAmount);  
  
  
 }  
 }  
}

package taskoops;  
import java.util.Scanner;  
  
  
public class XYZ {  
 // Method to calculate total amount spent on all products  
 public static double calculateTotalAmount(Product[] products) {  
 double totalAmount = 0;  
 for (Product product : products) {  
 totalAmount += product.getPrice() \* product.getQuantity();  
 }  
 return totalAmount;  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 // Array to store 5 products  
 Product[] products = new Product[5];  
  
 // Accept 5 product details from the user  
 for (int i = 0; i < 5; i++) {  
 System.*out*.println("Enter details for product " + (i + 1) + " (pid, price, quantity):");  
 int pid = scanner.nextInt();  
 double price = scanner.nextDouble();  
 int quantity = scanner.nextInt();  
  
 // Create a product and add it to the array  
 products[i] = new Product(pid, price, quantity);  
 }  
  
 // Find the product ID (pid) with the highest price  
 int maxPricePid = products[0].getPid();  
 double maxPrice = products[0].getPrice();  
  
 for (Product product : products) {  
 if (product.getPrice() > maxPrice) {  
 maxPrice = product.getPrice();  
 maxPricePid = product.getPid();  
 }  
 }  
  
 System.*out*.println("Product ID with the highest price: " + maxPricePid);  
  
 // Calculate and display the total amount spent  
 double totalAmount = *calculateTotalAmount*(products);  
 System.*out*.println("Total amount spent on all products: " + totalAmount);  
  
  
 }  
}

**Output:**

**Enter details for product 1 (pid, price, quantity):**

**1**

**20**

**2**

**Enter details for product 2 (pid, price, quantity):**

**2**

**1**

**4**

**Enter details for product 3 (pid, price, quantity):**

**3**

**8**

**3**

**Enter details for product 4 (pid, price, quantity):**

**4**

**6**

**8**

**Enter details for product 5 (pid, price, quantity):**

**5**

**7**

**4**

**Product ID with the highest price: 1**

**Total amount spent on all products: 144.0**

**1.3.Solution**

public class Account {

// Data member

private double balance;

// No-argument constructor

public Account() {

this.balance = 0; // Initialize balance to 0

}

// Parameterized constructor

public Account(double initialBalance) {

if (initialBalance >= 0) {

this.balance = initialBalance;

} else {

System.out.println("Initial balance cannot be negative. Setting balance to 0.");

this.balance = 0;

}

}

// Method to deposit the amount

public void deposit(double amount) {

if (amount > 0) {

balance += amount;

System.out.println("Deposited: " + amount);

} else {

System.out.println("Deposit amount must be positive.");

}

}

// Method to withdraw the amount

public void withdraw(double amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

System.out.println("Withdrawn: " + amount);

} else if (amount > balance) {

System.out.println("Insufficient balance.");

} else {

System.out.println("Withdrawal amount must be positive.");

}

}

// Method to display the balance

public void displayBalance() {

System.out.println("Current balance: " + balance);

}

// Main method to demonstrate functionality

public static void main(String[] args) {

// Create an account using no-argument constructor

Account account1 = new Account();

account1.displayBalance();

// Create an account using parameterized constructor

Account account2 = new Account(1000);

account2.displayBalance();

// Perform deposit, withdrawal, and display operations

account2.deposit(500); // Deposit money

account2.withdraw(300); // Withdraw money

account2.withdraw(2000); // Attempt to overdraw

account2.displayBalance(); // Display the balance

}

}

**OutPut:**

Current balance: 0.0

Current balance: 1000.0

Deposited: 500.0

Withdrawn: 300.0

Insufficient balance.

Current balance: 1200.0

**1.4 Solution**

package task\_1\_4;  
  
public class Person {  
 String name;  
 int age;  
}

package task\_1\_4;  
  
public class Employee extends Person{  
 int employeeID;  
 double salary;  
 public Employee(String name,int age){  
 System.*out*.println("Employee name is"+(super.name=name));  
 System.*out*.println("Employee age is"+(super.age=age));  
 }  
  
 public static void main(String[] args) {  
 Employee em=new Employee("Ramesh",20);  
 }  
  
}