**Instructions**

1.A person wants to create a Bank account, so he goes to Bank and the Bank requires following details to create a Bank account. Details such as first Name, last Name, age and address (Note: A descent address should comprise of Name, Door No, Street, City and Postal Code).

  Having given that the Bank would create an account with following details: Account Number and the Bank would also request the person to maintain a minimum balance of Rs.500, which he has to deposit for his account to get activated.

Here is a possible solution for the problem:

// A class that represents a bank account public class BankAccount {

// Fields to hold the account details private String firstName; private String lastName; private int age; private Address address; private String accountNumber; private double balance;

// A constant to hold the minimum balance requirement public static final double MIN\_BALANCE = 500.0;

// A constructor that takes the first name, last name, age, address, and account number as parameters and initializes the fields public BankAccount(String firstName, String lastName, int age, Address address, String accountNumber) { this.firstName = firstName; this.lastName = lastName; this.age = age; this.address = address; this.accountNumber = accountNumber; this.balance = 0.0; // Initially, the balance is zero }

// Getter methods for the fields public String getFirstName() { return firstName; }

public String getLastName() { return lastName; }

public int getAge() { return age; }

public Address getAddress() { return address; }

public String getAccountNumber() { return accountNumber; }

public double getBalance() { return balance; }

// Setter methods for the fields public void setFirstName(String firstName) { this.firstName = firstName; }

public void setLastName(String lastName) { this.lastName = lastName; }

public void setAge(int age) { this.age = age; }

public void setAddress(Address address) { this.address = address; }

// A method that deposits a given amount to the account and returns true if successful, false otherwise public boolean deposit(double amount) { // Check if the amount is positive if (amount > 0) { // Add the amount to the balance balance += amount; // Return true to indicate success return true; } else { // Return false to indicate failure return false; } }

// A method that withdraws a given amount from the account and returns true if successful, false otherwise public boolean withdraw(double amount) { // Check if the amount is positive and does not exceed the balance and the minimum balance requirement if (amount > 0 && balance - amount >= MIN\_BALANCE) { // Subtract the amount from the balance balance -= amount; // Return true to indicate success return true; } else { // Return false to indicate failure return false; } } }

// A class that represents an address public class Address {

// Fields to hold the address details private String name; private String doorNo; private String street; private String city; private String postalCode;

// A constructor that takes the name, door number, street, city, and postal code as parameters and initializes the fields public Address(String name, String doorNo, String street, String city, String postalCode) { this.name = name; this.doorNo = doorNo; this.street = street; this.city = city; this.postalCode = postalCode; }

// Getter methods for the fields public String getName() { return name; }

public String getDoorNo() { return doorNo; }

public String getStreet() { return street; }

public String getCity() { return city; }

public String getPostalCode() { return postalCode; }

}

// A class that demonstrates the BankAccount class import java.util.Scanner; // To get user input

public class UseBankAccount {

// The main method that runs the program public static void main(String[] args) {

// Create a Scanner object to get user input

Scanner sc = new Scanner(System.in);

// Declare variables to store the user input

String firstName, lastName, name, doorNo, street, city, postalCode, accountNumber;

int age;

// Ask the user to enter their first name

System.out.print("Enter your first name: ");

firstName = sc.nextLine(); // Read the input as a string

// Ask the user to enter their last name

System.out.print("Enter your last name: ");

lastName = sc.nextLine(); // Read the input as a string

// Ask the user to enter their age

System.out.print("Enter your age: ");

age = sc.nextInt(); // Read the input as an int

// Consume the remaining newline

sc.nextLine();

// Ask the user to enter their address name

System.out.print("Enter your address name: ");

name = sc.nextLine(); // Read the input as a string

// Ask the user to enter their door number

System.out.print("Enter your door number: ");

doorNo = sc.nextLine(); // Read the input as a string

// Ask the user to enter their street

System.out.print("Enter your street: ");

street = sc.nextLine(); // Read the input as a string

// Ask the user to enter their city

System.out.print("Enter your city: ");

city = sc.nextLine(); // Read the input as a string

// Ask the user to enter their postal code

System.out.print("Enter your postal code: ");

postalCode = sc.nextLine(); // Read the input as a string

// Ask the user to enter their account number

System.out.print("Enter your account number: ");

accountNumber = sc.nextLine(); // Read the input as a string

// Close the scanner object

sc.close();

// Create an instance of the Address class with the user input

Address address = new Address(name, doorNo, street, city, postalCode);

// Create an instance of the BankAccount class with the user input and the address object

BankAccount ba = new BankAccount(firstName, lastName, age, address, accountNumber);

// Display a welcome message

System.out.println("Welcome " + ba.getFirstName() + " " + ba.getLastName() + "!");

// Display the account details

System.out.println("Your account details are:");

System.out.println("Account number: " + ba.getAccountNumber());

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

System.out.println("Address: " + ba.getAddress().getName() + ", " + ba.getAddress().getDoorNo() + ", "

+ ba.getAddress().getStreet() + ", " + ba.getAddress().getCity() + ", " + ba.getAddress().getPostalCode());

// Display the current balance

System.out.println("Your current balance is: Rs." + ba.getBalance());

// Deposit Rs.1000 to activate the account

System.out.println("Depositing Rs.1000 to activate your account...");

if (ba.deposit(1000.0)) {

System.out.println("Deposit successful!");

} else {

System.out.println("Deposit failed!");

}

// Display the updated balance

System.out.println("Your updated balance is: Rs." + ba.getBalance());

// Withdraw Rs.200 from the account

System.out.println("Withdrawing Rs.200 from your account...");

if (ba.withdraw(200.0)) {

System.out.println("Withdrawal successful!");

} else {

System.out.println("Withdrawal failed!");

}

// Display the final balance

System.out.println("Your final balance is: Rs." + ba.getBalance());

} }

2.What is object and class in Java with example?

An object is a real-world entity that has some state and behavior. For example, a person, a car, a book, etc. are objects. An object can be physical or logical, depending on the context.

A class is a blueprint or template that defines the common properties and methods of a group of objects. A class is a logical entity that does not exist in the real world. A class can be used to create multiple objects of the same type.

For example, we can create a class called Person that has properties such as name, age, gender, and methods such as walk, talk, eat, etc. The class Person does not represent any specific person, but it defines the general characteristics of a person.

We can create objects of the class Person by using the new keyword and assigning them to reference variables. For example,

Person p1 = new Person(); // creates an object of class Person and assigns it to p1 Person p2 = new Person(); // creates another object of class Person and assigns it to p2

Each object has its own copy of the properties and methods defined by the class. We can access and modify the properties and methods of an object by using the dot (.) operator. For example,

p1.name = “John”; // sets the name property of p1 to “John” p2.age = 25; // sets the age property of p2 to 25 p1.walk(); // calls the walk method of p1 p2.talk(); // calls the talk method of p2

Objects are also known as instances of a class because they are created from the class definition. A class can have many instances or objects, but an object can belong to only one class. Objects are also called runtime entities because they are created and destroyed during the execution of the program.

3.What are classes and Objects?

Classes and objects are basic concepts of object-oriented programming that revolve around real-life entities. A class is a **blueprint** or **prototype** that defines the common **attributes** and **methods** of a group of objects. [An object is an **instance** of a class that has a specific **state**, **behavior**, and **identity**1](https://www.geeksforgeeks.org/classes-objects-java/)[2](https://www.w3schools.com/java/java_classes.asp).

For example, in Java, you can create a class named Student that has attributes like id and name, and methods like study and takeExam. Then you can create objects of the Student class, such as s1, s2, etc., that have different values for the attributes and can invoke the methods[1](https://www.geeksforgeeks.org/classes-objects-java/).

Bottom of Form

4.What is the use of ‘this’ keyword?

[The this keyword in Java is a **reference variable** that refers to the **current object** of a class1](https://www.javatpoint.com/this-keyword)[2](https://www.w3schools.com/java/ref_keyword_this.asp). It can be used for various purposes, such as:

* [To refer to the current class **instance variable** when there is a conflict with a local variable or a parameter with the same name1](https://www.javatpoint.com/this-keyword)[2](https://www.w3schools.com/java/ref_keyword_this.asp).
* [To invoke the current class **method** (implicitly) or **constructor** (explicitly) within the same class1](https://www.javatpoint.com/this-keyword)[3](https://www.c-sharpcorner.com/article/this-keyword-in-java/).
* [To pass the current object as an **argument** in a method call or a constructor call1](https://www.javatpoint.com/this-keyword)[4](https://www.studytonight.com/java/this-keyword-in-java.php).
* To return the current object from a **method**[1](https://www.javatpoint.com/this-keyword)[4](https://www.studytonight.com/java/this-keyword-in-java.php).

For example, in the following code, the this keyword is used to refer to the instance variable x and to invoke the constructor Main(int x):

public class Main {

int x;

public Main() {

this(10); // calling constructor with this

}

public Main(int x) {

this.x = x; // referring instance variable with this

}

public static void main(String[] args) {

Main obj1 = new Main();

Main obj2 = new Main(5);

System.out.println(obj1.x); // prints 10

System.out.println(obj2.x); // prints 5

}

}

5.What is the super() function?

[The super() function in Java is used to **invoke** the **constructor** of the **parent class**1](https://www.javatpoint.com/super-keyword)[2](https://www.geeksforgeeks.org/super-keyword/). [It is usually used when the subclass wants to initialize some common state or behavior from the parent class3](https://stackoverflow.com/questions/3767365/super-in-java).

For example, in the following code, the super() function is used to call the constructor of the Animal class from the Dog class:

class Animal {

String name;

Animal(String name) {

this.name = name;

}

}

class Dog extends Animal {

String breed;

Dog(String name, String breed) {

super(name); // calling parent class constructor

this.breed = breed;

}

}

6.Can a top-level class be private or protected?

[No, we cannot declare a top-level class as private or protected1](https://www.tutorialspoint.com/can-we-declare-a-top-level-class-as-protected-or-private-in-java)[2](https://net-informations.com/java/cjava/private.htm). It can be either public or default (no modifier). [If it does not have a modifier, it is supposed to have a default access, which means it is visible only within its own package3](https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html).

[A top-level class as private or protected would be completely useless because nothing would have access to it outside the class itself3](https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html)[2](https://net-informations.com/java/cjava/private.htm). [The compiler will complain that the modifier private or protected is not allowed here1](https://www.tutorialspoint.com/can-we-declare-a-top-level-class-as-protected-or-private-in-java)[4](https://stackoverflow.com/questions/1913863/why-cant-we-define-a-top-level-class-as-private).

Private or protected classes are allowed, but only as inner or nested classes. [In that case, the access is restricted to the scope of the outer class4](https://stackoverflow.com/questions/1913863/why-cant-we-define-a-top-level-class-as-private) .