Q1. Write a simple Banking System program by using OOPs concept where you can get account Holder name balance etc?

Ans : package com.bankingApplication;

import java.util.Scanner;

/\*

\* @author SURESH SENCHA

\*/

//user defined exception to generate error message if the user enters wrong credentials

class InvalidCredentialsException extends Exception{

private static final long serialVersionUID = 1L;

InvalidCredentialsException(String a){

super(a);

}

}

//class to check the balance in the account

class CheckBalance{

public void currentBalance(int currentBalance) {

System.out.println("your account balance is "+currentBalance);

System.out.println("Thankyou for using our application.");

}

}

//class to perform deposit operation on the account

class Deposit{

public int CurrentBalanceAndDepositedMoney(int currentBalance, int depositMoney) {

System.out.println("successfully "+depositMoney+" was Deposited to your account");

return currentBalance+depositMoney;

}

}

//class to perform the withdraw operation on the account

class WithDraw{

public int checkingBalanceToWithdraw(int currentBalance,int withDrawMoney) {

if(currentBalance>=withDrawMoney) {

System.out.println("successfully "+withDrawMoney+" was withdrawn from your account");

return currentBalance-withDrawMoney;

} else {

System.out.println("U dont have enough balance to withdraw, make sure "

+ "your current Balance is greater than withDrawMoney");

}

return currentBalance;

}

}

public class BankingApplication {

static Scanner scan=new Scanner(System.in);

public void performingBankingApplicatio() {

int currentBalance=20000;

String option="yes";

while(option.equalsIgnoreCase("yes")) {

System.out.println("enter whether u want to check your balance r deposit r withdraw or exit ");

String value=scan.next();

switch(value){

case "check":

new CheckBalance().currentBalance(currentBalance);

break;

case "deposit":

System.out.println("enter how much money u want to deposit");

int depositMoney=scan.nextInt();

currentBalance=new Deposit().CurrentBalanceAndDepositedMoney(currentBalance, depositMoney);

System.out.println("now your balance after deposit is "+currentBalance);

break;

case "withdraw":

System.out.println("enter how much money you want to withdraw");

int withDrawMoney=scan.nextInt();

int ModifiedCurrentBalance=new WithDraw().checkingBalanceToWithdraw(currentBalance, withDrawMoney);

if(ModifiedCurrentBalance<currentBalance) {

System.out.println("now your balance after withdraw is "+ModifiedCurrentBalance);

}

else {

System.out.println("your current balance is "+ModifiedCurrentBalance);

}

break;

case "exit":

System.out.println("Successufully loged out from the application");

System.exit(0);

break;

default:

System.out.println("enter proper value to work on your account :(");

}

System.out.println("enter yes if u wnt to use the application again");

option=scan.next();

}

if(option.equalsIgnoreCase("yes")!=true) {

System.out.println("application terminated...");

}

}

public static void checkingCredentials() throws InvalidCredentialsException {

final String Name="12345@ybl";

final String pwd="12345";

System.out.println("enter userName");

String userName=scan.next();

System.out.println("enter password");

String password=scan.next();

if(Name.equals(userName) && pwd.equals(password)) {

new BankingApplication().performingBankingApplicatio();

}else {

throw new InvalidCredentialsException("temperarly your account blocked, you entered wrong credentils too many times ");

}

}

public static void main(String[] args) throws InterruptedException {

System.out.println("welcome...! please enter your login details to access our services ");

Thread.currentThread().sleep(1000);

try {

BankingApplication.checkingCredentials();

}catch(Exception e) {

try {

System.out.println("credentials mismatched please enter again");

Thread.currentThread().sleep(1000);

BankingApplication.checkingCredentials();

}catch(Exception e1) {

try {

System.out.println("credentials mismatched please enter again");

Thread.currentThread().sleep(1000);

BankingApplication.checkingCredentials();

}catch(Exception e2) {

System.out.println(e2.getMessage());

}

}

}

}

}

Q2. Write a Program where you inherit method from parent class and show method Overridden Concept?

Ans : This is a Java Program to Show Method Overriding in a Class Using Inheritance Class.

In a class hierarchy, when a method in a subclass has the same name and type signature as a method in its superclass, then the method in the subclass is said to override the method in the superclass. Here we made showme() method in both classes and achieve the concept of Method Overriding.

Here is the source code of the Java Program to Show Method Overriding in a Class Using Inheritance Class. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

1. class Base
2. {
3. void showme()
4. {
5. System.out.println(" Base class method");
6. }
7. }
8. class Child extends Base
9. {
10. void showme()
11. {
12. System.out.println("Child class method");
13. }
14. public static void main(String... a)
15. {
16. Child obj = new Child();
17. obj.showme();
18. }
19. }

Q3. Write a program to show run time polymorphism in java?

Ans :

class Animal {

   public void move() {

      System.out.println("Animals can move");

   }

}

class Dog extends Animal {

   public void move() {

      System.out.println("Dogs can walk and run");

   }

}

public class TestDog {

   public static void main(String args[]) {

      Animal a = new Animal(); // Animal reference and object

      Animal b = new Dog(); // Animal reference but Dog object

      a.move(); // runs the method in Animal class

      b.move(); // runs the method in Dog class

   }

}

Q4.Write a program to show Compile time polymorphism in java?

Ans :

public class MethodOverloading {

void show(int num1){

System.out.println("number 1 : " + num1);

}

void show(int num1, int num2){

System.out.println("number 1 : " + num1+ " number 2 : " + num2);

}

public static void main(String[] args){

MethodOverloading obj = new MethodOverloading();

obj.show(3);

obj.show(4, 5);

}

}

Q5. Achieve loose coupling in java by using OOPs  concept?

Ans : Loose coupling - When an object gets the object to be used from the outside, then it is a loose coupling situation. As the main object is merely using the object, this object can be changed from the outside world easily marked it as loosely coupled objects

Q6. What is the benefit of encapsulation in java?

Ans : Benefits of encapsulation programming

* Hiding data: Users will have no idea how classes are being implemented or stored. ...
* More flexibility: Enables you to set variables as red or write-only. ...
* Easy to reuse: With encapsulation, it's easy to change and adapt to new requirements.

Q7.Is java a t 100% Object oriented Programming language? If no why ?

Ans : Java is not 100% objects-oriented because of the existence of primitive data types, use of static keywords and wrapper classes. Object-oriented programming is a methodology that organizes software designs around objects rather than functions and logic.

Q8.What are the advantages of abstraction in java?

Ans : Advantages of Abstract Classes

* Abstract class in Java is highly beneficial in writing shorter codes.
* Abstraction in Java avoids code duplication.
* Abstract classes enable code reusability.
* Changes to internal code implementation are done without affecting classes.

Q9.What is an abstraction explained with an Example?

Ans : Abstraction is a process of hiding the implementation details and showing only functionality to the user. Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message.

class abstract Bird {

String name;

int size;

int strength;

protected abstract void attack();

public String getName(){

return name;

}

}

Q10.What is the final class in Java?

Ans : A class can be made final by using the final keyword. The final class cannot be inherited and so the final keyword is commonly used with a class to prevent inheritance.