**MD ARBAZ KHAN**

**16928**

1. Create a class with a method that prints "This is parent class" and its subclass with another method that prints "This is child class". Now, create an object for each of the class and call
   1. - method of parent class by object of parent class
   2. - method of child class by object of child class

Ans –

public class parent {  
 parent(){  
 System.*out*.println("This is a Parent Class");  
 }  
}

public class child extends parent{  
 child(){  
 System.*out*.println("This is a Child Class");  
 }  
}

public class Main {  
 public static void main(String[] args) {  
 child c = new child();  
 }  
}

1. In the above example, declare the method of the parent class as private and then repeat the operations.

public class parent {  
 private parent(){  
 System.*out*.println("This is a Parent Class");  
 }  
}

public class child extends parent{  
 child(){  
 System.*out*.println("This is a Child Class");  
 }  
}

public class Main {  
 public static void main(String[] args) {  
 child c = new child();  
 }  
}

**AN ERROR WILL OCCUR IN THIS PROGRAM BECAUSE PARENT CLASS IS INHERITED IN CHILD CLASS AND WE ARE CALLING IT THROUGH CHILD CLASS**

1. With the help of a demo code, explain the constructor calling sequence in multi-level inheritance.

Ans – public class parent {  
 private parent(){  
 System.*out*.println("This is a Parent Class");  
 }  
}

public class child extends parent{  
 child(){  
 System.*out*.println("This is a Child Class");  
 }  
}

public class Main {  
 public static void main(String[] args) {  
 child c = new child();  
 }  
}

public class childChild extends child {  
 childChild(){  
 System.*out*.println("This is a child's Child class");  
 }  
}

public class Main {  
 public static void main(String[] args) {  
 childChild c = new childChild();  
 }  
}

1. Create a class 'Parent' with a method 'message'. It has two subclasses each having a method with the same name 'message' that prints "This is first subclass" and "This is second subclass" respectively. Call the methods 'message' by creating an object for each subclass.

Ans –

public class parent {  
  
}

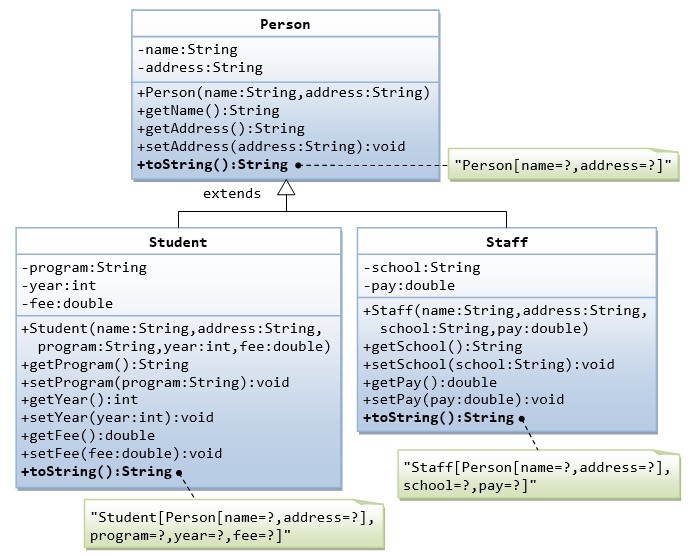
public class child1 extends parent{  
 public void method(){  
 System.*out*.println("This is a first sub Class");  
 }  
}

public class child2 extends parent {  
 public void method(){  
 System.*out*.println("This is a second sub class");  
 }  
}

public class Main {  
 public static void main(String[] args) {  
 child1 c1 = new child1();  
 c1.method();  
 child2 c2 = new child2();  
 c2.method();  
 }  
}

1. Make the class ‘Parent’ as an abstract class and the method ‘message’ as abstract. Override the method in both subclasses. Call the methods ‘message’ by creating the reference of Parent class and storing the object of each subclass in it. Comment the findings which method got invoked when.

Ans –

1. Write the classes as shown in the following class diagram. 

1. Mark all the overridden methods with annotation @Override.
2. Call the constructor of Person class from Student & Staff class constructor using super keyword.
3. In the overridden method, use super keyword to call the base class method.

Ans –

**PERSON**

public class person {  
  
 public String name;  
 public String address;  
  
 public String getName() {  
 return name;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 @Override  
 public String toString() {  
 return "student{" +  
 "name='" + name +  
 ", address='" + address +  
 '}';  
 }  
  
 public void person(String name, String address){  
 this.name = name;  
 this.address = address;  
 }  
  
}

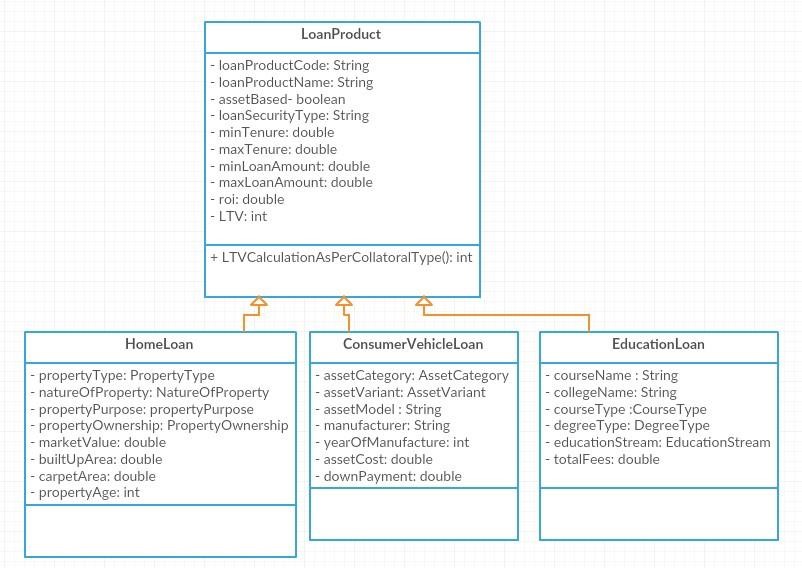
**STUDENT**

public class student extends person{  
 private String program;  
 private int year;  
 private double fee;  
 person p = new person();  
 public student(String name, String address, String program, int year, double fee) {  
 this.program = program;  
 this.year = year;  
 this.fee = fee;  
 p.name = name;  
 p.setAddress(address);  
 }  
  
 public String getProgram() {  
 return program;  
 }  
  
 public void setProgram(String program) {  
 this.program = program;  
 }  
  
 public int getYear() {  
 return year;  
 }  
  
 public void setYear(int year) {  
 this.year = year;  
 }  
  
 public double getFee() {  
 return fee;  
 }  
  
 public void setFee(double fee) {  
 this.fee = fee;  
 }  
  
 @Override  
 public String toString() {  
 return "student[" +  
 "program='" + program +  
 ", year=" + year +  
 ", fee=" + fee +  
 ", p=" + p +  
 ']';  
 }  
  
 public void print() {  
 super.person(p.name, p.address);  
 super.person(name, address);  
  
 }  
}

**STAFF**

public class staff extends person {  
 private String school;  
 private double pay;  
 person p = new person();  
  
 public staff(String name, String address, String school, double pay) {  
 p.name = name;  
 p.setAddress(address);  
 this.school = school;  
 this.pay = pay;  
 }  
  
 public String getSchool() {  
 return school;  
 }  
  
 public void setSchool(String school) {  
 this.school = school;  
 }  
  
 public double getPay() {  
 return pay;  
 }  
  
 public void setPay(double pay) {  
 this.pay = pay;  
 }  
  
 @Override  
 public String toString() {  
 return "staff{" +  
 "school='" + school +  
 ", pay=" + pay +  
 ", p=" + p +  
 '}';  
 }  
 person p1 = new person();  
 public void print() {  
 super.person(p1.name, p1.address);  
 super.person(name, address);  
 }  
  
}

* 1. Create the LoanProduct class hierarchy according to the given Class Diagram.



Create the required constructor and write a test class to test the functionalities.

* + 1. The method ‘LTVCalculationAsPerCollatoralType’ need to be overridden in all three classes. The formula to be used is given in Assignment-1. Pass the ‘LoanAmountAsked’ value in the method.
    2. Use upcasting to call this method.

Ans –

import Assignment\_1.Q7;  
public class loanProduct {  
 private String loanProductCode;  
 private String loanProductName;  
 private boolean assetBased;  
 private String loanSecurityType;  
 private double minTenure;  
 private double maxTenure;  
 private double minLoanAmount;  
 private double maxLoanAmount;  
 private double roi;  
 private int ltv;  
  
 public int LTVCalculationAsPerCollateralType(){  
 Q7 q7 = (Q7)new loanProduct();  
 return q7;  
 }  
}

import Assignment\_1.Q7;  
  
public class HomeLoan<PropertyType, natureOfProperty, propertyPurpose, propertyOwnership> extends loanProduct {  
 private PropertyType propertyType;  
 private natureOfProperty natureOfProperty;  
 private propertyPurpose propertyPurpose;  
 private propertyOwnership propertyOwnership;  
  
 private double marketValue;  
 private double builtUpArea;  
 private double carpetArea;  
 private int propertyAge;  
  
 public int LTVCalculationAsPerCollateralType(){  
 Q7 q7 = (Q7)new HomeLoan();  
 return 0;  
 }  
}

import Assignment\_1.Q7;  
  
public class ConsumerVehicleLoan<assetCategory, assetVariant> extends loanProduct {  
 private assetCategory assetCategory;  
 private assetVariant assetVariant;  
  
 private String assetModel;  
 private String manufacturer;  
 private int yearOfManufacture;  
 private double assetCost;  
 private double downPayment;  
  
 public int LTVCalculationAsPerCollateralType(){  
 Q7 q7 = (Q7)new loanProduct();  
 return 0;  
 }  
}

import Assignment\_1.Q7;  
  
public class EducationLoan<educationStream, degreeType, courseType>extends loanProduct {  
 private String courseName;  
 private String collegeName;  
  
 private courseType courseType;  
 private degreeType degreeType;  
  
 private educationStream educationStream;  
  
 private double totalFees;  
  
 public int LTVCalculationAsPerCollateralType(){  
 Q7 q7 = (Q7)new EducationLoan();  
 return 0;  
 }  
}

* 1. Make the base class – LoanProduct as an abstract class and the method as an abstract method. Use upcasting to call the abstract method.

Ans –

public abstract class loanProduct {  
 private String loanProductCode;  
 private String loanProductName;  
 private boolean assetBased;  
 private String loanSecurityType;  
 private double minTenure;  
 private double maxTenure;  
 private double minLoanAmount;  
 private double maxLoanAmount;  
 private double roi;  
 private int ltv;  
  
  
  
 public abstract int LTVCalculationAsPerCollateralType();  
}

public class HomeLoan<PropertyType, natureOfProperty, propertyPurpose, propertyOwnership> extends loanProduct {  
 private PropertyType propertyType;  
 private natureOfProperty natureOfProperty;  
 private propertyPurpose propertyPurpose;  
 private propertyOwnership propertyOwnership;  
  
 private double marketValue;  
 private double builtUpArea;  
 private double carpetArea;  
 private int propertyAge;  
  
 public int LTVCalculationAsPerCollateralType(){  
 loanProduct lphl= (loanProduct) new HomeLoan<>();  
 return 0;  
 }  
}

public class ConsumerVehicleLoan<assetCategory, assetVariant> extends loanProduct {  
 private assetCategory assetCategory;  
 private assetVariant assetVariant;  
  
 private String assetModel;  
 private String manufacturer;  
 private int yearOfManufacture;  
 private double assetCost;  
 private double downPayment;  
  
 public int LTVCalculationAsPerCollateralType(){  
 loanProduct lpcvl = (loanProduct) new ConsumerVehicleLoan<>();  
 return 0;  
 }  
}

public class EducationLoan<educationStream, degreeType, courseType>extends loanProduct {  
 private String courseName;  
 private String collegeName;  
  
 private courseType courseType;  
 private degreeType degreeType;  
  
 private educationStream educationStream;  
  
 private double totalFees;  
  
 public int LTVCalculationAsPerCollateralType(){  
 loanProduct lpedl = (loanProduct) new EducationLoan<>();  
 return 0;  
 }  
}