Q1/Assume that we change the CreditCard class (see Code Fragment 1.5) so that instance variable balance has private visibility. Why is the following implementation of the PredatoryCreditCard.charge method flawed? public boolean charge(double price) { boolean isSuccess = super.charge(price); if (!isSuccess) charge(5); // the penalty return isSuccess; }

الحل: يدخل بدواره لانهائيه لانه ايوجد استدعاء ذاتي

Q2/ Assume that we change the CreditCard class (see Code Fragment 1.5) so that instance variable balance has private visibility. Why is the following implementation of the PredatoryCreditCard.charge method flawed? public boolean charge(double price) { boolean isSuccess = super.charge(price); if (!isSuccess) super.charge(5); // the penalty return isSuccess; }

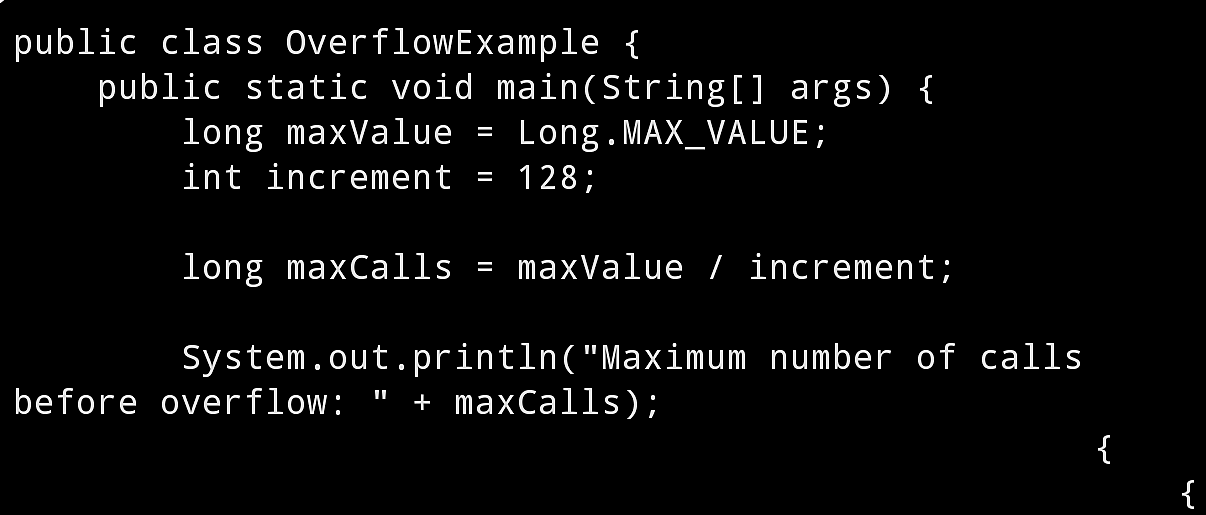
الحل: اذكان الليمت قريب لل5 فان صاحب البطاقه ينجو من العقوبه .

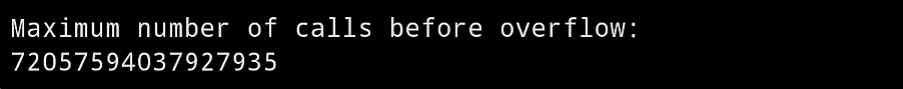
Q3/ Give a short fragment of Java code that uses the progression classes from Section 2.2.3 to find the eighth value of a Fibonacci progression that starts with 2 and 2 as its first two values

الحل : مرفق مع الملف ......

Q4/ If we choose an increment of 128, how many calls to the nextValue method from the ArithmeticProgression class of Section 2.2.3 can we make before we cause a long-integer overflow?

الحل:





Q8/Can two interfaces mutually extend each other? Why or why not?

الحل : لايسمح بالوراثه العكسيه لانها تكون وراثه دائريه ولامعنى لهذا الوراثه .

Q9/ What are some potential efficiency disadvantages of having very deep inheritance trees, that is, a large set of classes, A, B, C, and so on, such that B extends A, C extends B, D extends C, etc.?

الحل: قد يؤدي الى زيادة التعقيد ويصعب صيانته يصبح فهم الكلاسات وتعديلها وقد يؤدي الى صعوبة اعادة الاستخدام.

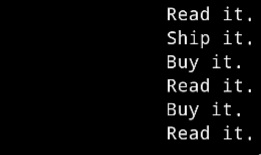
Q10/ What are some potential efficiency disadvantages of having very shallow inheritance trees, that is, a large set of classes, A, B, C, and so on, such that all of these classes extend a single class, Z?

الحل :

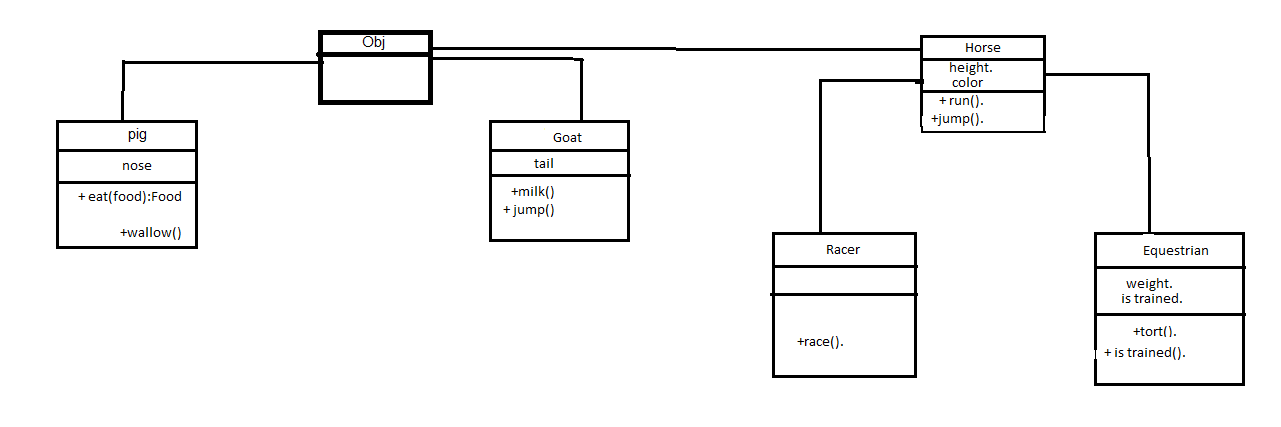
* Limited Abstraction.
* Increased Coupling.
* Reduced Reusability
* Inflexibility in Specialization.
* Overhead in initialization and memory usage.

Q11/ Consider the following code fragment, taken from some package: public class Maryland extends State { Maryland( ) { /∗ null constructor ∗/ } public void printMe( ) { System.out.println("Read it."); } public static void main(String[ ] args) { Region east = new State( ); State md = new Maryland( ); Object obj = new Place( ); Place usa = new Region( ); md.printMe( ); east.printMe( ); ((Place) obj).printMe( ); obj = md; ((Maryland) obj).printMe( ); obj = usa; ((Place) obj).printMe( ); usa = md; ((Place) usa).printMe( ); } } class State extends Region { State( ) { /∗ null constructor ∗/ } public void printMe( ) { System.out.println("Ship it."); } } class Region extends Place { Region( ) { /∗ null constructor ∗/ } public void printMe( ) { System.out.println("Box it."); } } class Place extends Object { Place( ) { /∗ null constructor ∗/ } public void printMe( ) { System.out.println("Buy it."); } } What is the output from calling the main( ) method of the Maryland class?

الحل :



Q12/ Draw a class inheritance diagram for the following set of classes: • Class Goat extends Object and adds an instance variable tail and methods milk( ) and jump( ). • Class Pig extends Object and adds an instance variable nose and methods eat(food) and wallow( ). • Class Horse extends Object and adds instance variables height and color, and methods run( ) and jump( ). • Class Racer extends Horse and adds a method race( ). • Class Equestrian extends Horse and adds instance variable weight and isTrained, and methods trot( ) and isTrained( )?

الحل: 

Q13/Consider the inheritance of classes from Exercise R-2.12, and let d be an object variable of type Horse. If d refers to an actual object of type Equestrian, can it be cast to the class Racer? Why or why not?

الحل: نعم يمكن ارساله

لانه اذا كان d يشير الى Equestrian وبما ان ال Racer هو فئه فرعيه من ال Horse والتي تشمل ال Equestrianأيضا فيمكننا ارسالها بأمان.

Q14/ Give an example of a Java code fragment that performs an array reference that is possibly out of bounds, and if it is out of bounds, the program catches that exception and prints the following error message: “Don’t try buffer overflow attacks in Java!?

الحل: في الملف المرفق......

Q15/ If the parameter to the makePayment method of the CreditCard class (see Code Fragment 1.5) were a negative number, that would have the effect of raising the balance on the account. Revise the implementation so that it throws an IllegalArgumentException if a negative amount is sent as a parameter.

public void makePayment(double amount) { // make a

payment if(amount<0) throw new IllegalArgumentException("Negative Amount is not Allowed"); balance -= amount; }

الحل: