Exercises and Homework

|  |  |
| --- | --- |
| R-2.4 | 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;  }  بسبب تكرار الدالة واستدعاء نفسها بشكل لا نهائي والحل استخدام كود supar The PredatoryCreditCard.charge method ecurse indefinitely |
| R-2.5 | 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;  }  تحاول عبارة super.charge (5) في التنفيذ المعيب تعديل متغير التوازن للفئة الفائقة مباشرة ، وهو أمر غير مسموح به بسبب رؤيته الخاصة. n either case, you can't be charged a fee if you are close enough to the balance that the fee (of value 5) would exceed your limit. |
| R-2.6 | 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  Import java.math.BigInteger;  Public class FIDONACCIPROG{  Public static viod main(String[] args){  int n=8;    Fibonacciprogression Fibonacci=new Fibonacciprogr ession(BigInteger.valueOf(2));  BigInteger eightValue= Fibonacci. GetNth Value(n);  System.out.println("the eighth value in the Fibonacci progression is:" +eighthValue);  } }    تتتن,2); fibonacci.printProgression(8 |
| R-2.7 | 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?  base  d on the fo before causing a long-integer overflow. |
| R-2.8 | Can two interfaces mutually extend each other? Why or why not?  لللللللللللليس من الممكن ان تقوم واجهتان بعمل وراثة لبعضهما لانه اذا تم تمديد الواجهتان بشكل مباشر سيؤدي ذلك الى أخطاء برمجية,hgbiguity and conflicts. Instead, interfaces can be used in c |
| R-2.9 | 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.?  \  استهلاك الذاكرة بشكل كبير  التعقيد وصعوبة الصيانة  تعقيد الاكواد وعدم فهمها  صعوبه التواصل وفك الشفرة  الوقت المستغرق يكون طويل بسبب التنقل |
| R-2.10 | 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?  استهلاك الذاكرة بشكل كبير التعقيد وصعوبة الصيانة الوقت المستغرق يكون طويل بسبب التنقل اداء البحث والاستعلام تعقيد الاكواد |
| R-2.11 | 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?  1-Read it.  2-Ship it.  3-Buy it.  4-Read it.  5-Box it.  6- Read it. |
| R-2.12 | 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( ).  Object  ^  |  Animal  ^  |  +------+-----------------+  | | |  Goat Pig Horse  | ^ ^  | | |  +-------------+--------+ |  | | | |  Racer Equestrian Pony WildHorse |
| R-2.13 | 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?  لايمكن تحويلها الى فئهRacer باستخدام العوامل المستدعاء  وليش لان Racer تمتد من فئه Horse وليس لها علاقه مباشره بفئه Equestrian عوامل الاسدعاه تستخدم الكائن الى فئات مشتقه من فئتها الاصليه في هذه الحااله Racerليست جزء من سلسه التوريث  *The answer is no because Racer is not sub or super for Equesrain Equestrian cannot be cast to class R\_2\_13.Racer (R\_2\_13.Equestrian and R\_2\_13.Racer are in unnamed module of loader 'app')* |
| R-2.14 | 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!”  try {  int[] numbers = {1, 2, 3};  // Attempt to access an element that is out of bounds  int value = numbers[5];  // This line will not execute if an exception is thrown  System.out.println("Value: " + value);  } catch (ArrayIndexOutOfBoundsException e) {  System.out.println("Don't try buffer overflow attacks in Java!");  } |
| R-2.15 | 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) {  if (amount < 0) {  throw new IllegalArgumentException("Payment amount cannot be negative");  }    balance -= amount;  }  الكود يقدم تعديلا على طريقهmakePaymentفي فئه creditcard ويقوم التعديل للتحقق من ان المبلغ سالبا لانه سيتم رفع الاستثمار على الحساب ولهذا يتم القاء اسثناء  IIIegalArgumentException  وهذا كود التعديل  Public viod makepayment (double amount){  If (amount<0){  IllegalArgumentException ("Nagative Amount is not Allowed");  }  Balance -=amount; |