**Data Structure Lab2 -Object-Oriented Design**

1-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;

}

هنا حينما يتغير balance الىprivate كلاس PredatoryCreditCard لن يستطيع الوصول المباشر اليه فحينما تفشل عملية super.charge(price); سيدخل ل IF وسوفي يتم استدعاء الدالة نفسها مما يسبب حلقة لانهائيه اذا استمر فشل super.charge(price); فالحل ان نجعله يستدعي الدالة الاصليه super.charge(5); فسوف يتم استعدائها مره واحدة

public boolean charge(double price) {

boolean isSuccess = super.charge(price);

if (!isSuccess)

super.charge(5); // the penalty

return isSuccess;

}

2- 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;

}

هنا خطا لانه عند استخدام super.charge(5) يحدث خرق للكبسله لانه يعدل لى balanceفلتجنب هذا نقوم باستخدام دوال للتعامل مع balanceيطريقة غير مباشرة

نضيف دالة لكلاس CreditCard

public boolean applyPenalty(double penaltyAmount) {

return charge(penaltyAmount);

}

public boolean charge(double price) {

boolean isSuccess = super.charge(price);

if (!isSuccess) {

boolean penaltySuccess= super. applyPenalty (5);

isSuccess= penaltySuccess;}

return isSuccess;

}

3-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.

public class FibonacciProgression extends Progression {

protected long prev;

public FibonacciProgression(long first, long second) {

super(first); // start with first value

prev = second-first; // second value

}

protected void advance() {

long temp = current;

current = current + prev;

prev = temp;

System.out.println(current);

}

public static void main(String[] args) {

FibonacciProgression prog = new FibonacciProgression(2, 2);

for (int i = 1; i <= 8; i++) {

prog.advance();

}

}

}

4-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?

الحد الاقصى ل long 9,223,370,036,854,775,807 الزيادة في كل الاستدعاء 128 فنقسم الحد الاقصى على 128 يطلع الناتج 72,057,594,037,927,935

5-Can two interfaces mutually extend each other? Why or why not?

لا لانه سوف يؤدي لحلقة توريث غير منتهيه وتعارض في العلاقات بين الواجهات وتعارض في تعريف الدوال

6-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.?

زيادة في التعقيد –تقليل من المرونه-زيادة في الوقت عند البحث عن الدوال-صعوبه في تطبيق التحسينات – زيادة استهلاك الذاكرة

7-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?

زيادة حجم كلاس الاب – وراثه لدوال غير ضرورية –قابليه التوسع والصيانه معقدة-التكرار في الوظائف-افتقار التخصيص

8-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?

run:

Read it.

Ship it.

Buy it.

Read it.

Box it.

Read it.

BUILD SUCCESSFUL (total time: 0 seconds)

9-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( ).

tail:String-

Goat

Pig

Horse

IS\_A

IS\_A

+milk():void

+jump():void

+eat():void

+wallow():void

+trot():void

+ isTrained ():void

- weight:int

-isTrained:boolean

+race():void

Equestrian

Racer

-height:int

-colort:String -

+run():void

+jump():void

nose:String-

10-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?

لا لا يمكن لانه class Racer و class Equestrian فرعيتان من class Horse فهما مستقلتان عن بعضهما ولاتوجد علاقة مباشرة بينهما

11-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!”

public class ArrayExample{

public static void main(String[] args){

int[] a={6,8,3,6};

try {

System.out.println(a[7]);

} catch (ArrayIndexOutOfBoundsException e){

System.out.println("Don’t try buffer overflow attacks in Java!");

}

}

}

12-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;  
 }