النظري حل التكليف

11.Consider the following code fragment, taken from some package: public class Maryland extends State { Maryland( ) { /∗ null constructor ∗/ } public void print Me( ) { System.out.println("Read it."); } public static void main(String[ ] rags) { Region east = new State( ); State md = new Maryland( ); Object obj = new Place( ); Place use = new Region( ); md.printMe( ); east.printMe( ); ((Place) obj).print Me( ); obj = md; ((Maryland) obj).print ME( ); obj = use; ((Place) obj).print ME( ); us = md; ((Place) USA).print Me( ); } } class State extends Region { State( ) { /∗ null constructor ∗/ } public void print ME( ) { System.out.println("Ship it."); } } class Region extends Place { Region( ) { /∗ null constructor ∗/ } public void print Me( ) { 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?

Output:

Read it.

Ship it.

Buy it.

Box it.

Read it.

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?

(عيوب الكفاءة للتصميم باستخدام شجرة وراثة مسطحة):

دون وجود مستويات وسيطة، فإن ذلك يسبب Z وكلها ترث مباشرة من فئة واحدة، C, B, A عندما يكون لديك عدد كبير من الفئات مثل عدة مشكال في الكفاءة والتصميم، منها:

1. تكرار الكود.

قد تحتاج إلى وظائف متشابهة، ولكن ألن كل فئة قائمة بذاتها، سيتم إعادة كتابة نفس الشيفرة في Z الفئات الفرعية التي ترث من الفئة عدة أماكن.

هذا يؤدي إلى صعوبة صيانة الكود وزيادة احتمالية الخطاء عند التعديل.

.2ثقل الفئة األساسية.

هي الفئة األساسية الوحيدة لجميع الفئات األخرى، قد تصبح عامة جًدا وتتضمن خصائص ووظائف ال تحتاجها جميع الفئات Z إذا كانت الفرعية.

هذا يسبب زيادة في تعقيد الفئة األساسية وإبطاء الأداء.

.3عدم وجود تخصص أو تنظيم.

شجرة الوراثة المسطحة تمنع إنشاء مستويات وسيطة لتنظيم الفئات بنا ًء على وظائف مشتركة أو متخصصة.

النتيجة: التصميم يصبح فوضوًيا وصعب التوسع، حيث ال يمكن الاستفادة من وراثة الوظائف المشتركة بشكل ف ّعال.

.4صعوبة التوسع والصيانة.

مع زيادة عدد الفئات الفرعية، يصبح من الصعب التحكم فيها وتعديلها إذا اعتمدت جميعها على الفئة األساسية.

قد يؤثر على جميع الفئات الفرعية، مما يزيد من مخاطر الأخطاء غير المتوقعة Z أي تعديل في الفئة األساسية.

.6ضعف إعادة استخدام الكود.

عدم وجود مستويات وسيطة في شجرة الوراثة يجعل من الصعب مشاركة السلوكيات أو الخصائص بين مجموعة محددة من الفئات.

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, etic?

عيوب الكفاءة للتصميم باستخدام شجرة وراثة عميقة جًدا.

.1صعوبة الفهم والصيانة.

.2زيادة الاعتماد المتسلسل.

.3ضعف الأداء.

.4تداخل السلوكيات والوظائف.

.5زيادة تعقيد التعديلات.

.7ضعف إعادة الاستخدام.

8.Can two interfaces mutually extend each other? Why or why not?

Two interfaces cannot mutually extend each other directly due to the potential for ambiguity and conflicts. Instead, interfaces can be used in conjunction with multiple inheritance to provide the desired functionality without introducing these issues

Cause Cyclic inheritance

الواجهات لا يمكن أن ترث من بعضها بشكل متبادل ألان ذلك يؤدي إلى وراثة دورية حيث تصبح كل، (Inheritance Cyclicواجهة تعتمد على الأخرى بشكل ال نهائي هذا يسبب مشكلتين رئيسيتين. (

1.غموض في تحديد العالقات.

المترجم ال يستطيع تحديد تسلسل الوراثة بشكل صحيح عند وجود دائرة، مما يؤدي إلى خطأ أثناء الترجمة.

تعارض محتمل في تعريف الدوال2

إذا كان للواجهتين دوال متشابهة بتعريفات مختلفة، فلن يستطيع النظام تحديد التعريف الذي يجب استخدامه، مما يؤدي إلى غموض وتصميم غير مستقر.

7.If we choose an increment of 128, how many calls to the next Value method from the Arithmetic Progression class of Section 2.2.3 can we make before we cause a long-integer overflow?

Arithmetic)) المتسلسلة الحسابية (class) في صف Value next لحساب عدد الاستدعاءات التي يمكن إجراؤها لدالة،long قبل حدوث تجاوز لقيمة النوع (Progression) نتبع الخطوات التالية:

المعطيات:

1. الزيادة. (Increment)
2. . نفترض أنها (Value Initial) القيمة الابتدائية.
3. . Java في long أقصى قيمة للنوع.

صيغة المتسلسلة الحسابية:

An = a0 + n .d

حيث:

(القيمة الابتدائية) تساوي0 هنا:

(الزيادة) تساوي 128 هنا:

n: عدد الخطوات

تجاوز الحد:

. يحدث التجاوز عندما:

an > 9,223,372,036,854,775,807

بما أن0 = a0 فإن:

n.128 > 9,223,372,036,854,775,807

حساب عدد الخطوات n:

n = 9,223,372,036,854,775,807

128

العملية الحسابية:

نقوم بالقسمة:

n = 72,057,594,037,927,936

يمكن استدعاء دالة next value 72,057,594,037,927,936 مرة قبل حدوث نجاوز القيمة النوع long

5.Assume that we change the Credit Card class (see Code Fragment 1.5) so that instance variable balance has private visibility.

Why is the following implementation of the Predatory CreditCard.charge method flawed?

public boolean charge(double price) {

boolean is Success = super. charge(price);

if (!isSuccess)

super.charge(5); // the penalty

return isSuccess;

الطريقة معيبة لأنها:

Balance-لا تحترم الخصوصية الخاصة للمتغير 1

تدمج منطقتين مختلفتين: شراء غرض وفرض غرامة -2

**لعيب في الطريقة هو انها تدمج بين منطق شراء المنتج وفرض الغرامة باستخدام نفس الطريقة ا -3**

**. مما يسبب تعقيدا في التعامل مع الرصيد Super.charge**

4.Assume that we change the Credit Card 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;

}

**معيبة لأنها تحتوي على استدعاء ذاتي غير محدود فعند فشل عملية PredatoryCreditCard.charge طريقة**

**. حلقة لا نهائية وتجاوز الحد الائتماني تستدعي نفسها لإضافة غرامة مما يؤدي الى**

يمكن استدعاء دالة next value 72,057,594,037,927,936 مرة قبل حدوث نجاوز القيمة النوع long

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?

. Racer إلى النوع d لا يمكن تحويل الكائن

وال توجد عالقة وراثة مباشرة أو غير مباشرة، Horse هما فئتان مستقلتان تشتقان من الفئة األساسية Racer Equestrian الفئتان. بينهما

. محاولة التحويل ستؤدي إلى خطأ أثناء التنفيذ ألن النظام ال يستطيع مطابقة النوعين

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 is Trained, and methods trot () and is Trained ().

Object

Dig

+milk ()

+jump ()

Goat

Is a

Is a

Is a

+ trot ()

+is trained ()

Equestrian

Is a

Is a

+race ()

Racer

Weight

is Trained

+run ()

+jump ()

Hours

Height

color

+eat(food)

+wallow

nose

rail