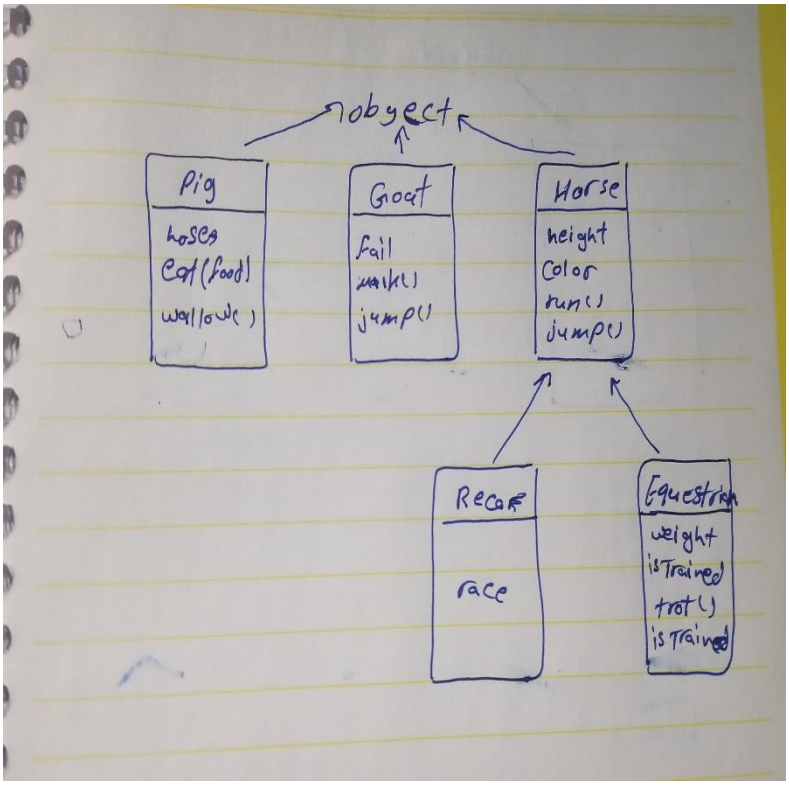
## Exercises and Homework

| | | |
|---|---|---|
| 1 | 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?<br>public boolean charge(double price) {<br>boolean isSuccess = super.charge(price);<br>if (!isSuccess)<br>charge(5); // the penalty<br>return isSuccess;<br>}<br><br>public boolean charge(double price) {<br>boolean isSuccess = super.charge(price);<br>if ( !isSuccess&& getBalance( ) >= 5) {<br>super.charge(5);<br>}<br>return isSuccess |
| 2 | R-2.5 | Assume that we change the CreditCard class (see Code Fragment 1.5) so that instance variable balance has private visibility.<br>Why is the following implementation of the PredatoryCreditCard.charge method flawed?<br>الحل الصحيح:<br><br>تطبيق الغرامة فقط إذا كانت لن تتجاوز الحد الائتماني باستخدام أدوات الوصول مثل: |

| | | |
|---|---|---|
| | | public boolean charge(double price) {<br>boolean isSuccess = super.charge(price);<br>if (!isSuccess) {<br>double newBalance = super.getBalance() + 5;<br>if (newBalance <= super.getCreditLimit()) {<br>super.charge(5); // فرض الغرامة فقط إذا كان ضمن الحد<br>}<br>}<br>return isSuccess;<br>} |
| 3 | 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.<br>class with a method getNextValue()<br>Fibonacci fibonacci = new Fibonacci(2, 2); // Initialize with first two values<br><br>for (int i = 0; i < 6; i++) { // Calculate the 8th value, starting from the 3rd<br>fibonacci.getNextValue();<br>}<br><br>int eighthValue = fibonacci.getNextValue();<br>System.out.println("The eighth value is: " + eighthValue); |
| 4 | R-2 | 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? |

| | | |
|---|---|---|
| | .7 | ArithmeticProgression progression = new ArithmeticProgression(0, 128);<br>long maxValue = Long.MAX_VALUE;<br>long currentValue = 0;<br>int count = 0;<br><br>while (currentValue < maxValue) {<br>currentValue = progression.getNextValue();<br>count++;<br>}<br><br>System.out.println("Number of calls before overflow: " + count); |
| 5 | R-2.8 | Can two interfaces mutually extend each other? Why or why not?<br>لا يمكن لواجهتين أن ترثان بعضهما البعض<br>بسبب:الوراثه تعني ان الفئة التي تنفذ الواجهه ترث جميع الطرق المجردة التي تم تعريفها في تلك الواجهه . و حدوث حلقه لا نهائيه سبب ان كل واجهه تحتاج الى تنفيذ الطرق الموجوده في الحلقه الأخرى |
| 6 | R-2 | 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.? |

| | | |
|---|---|---|
| | .9 | تعقيد الكود , صعوبة إضافة تعديلات ,تضخم الكود |
| 7 | 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?<br>صعوبة في التوسعه , قلة المرونه |
| 8 | 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?<br><br>Ship it<br>Box it<br>Buy it<br>Read it<br>Box it<br>Buy it |
| 9 | R-2. | 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 |

| | | |
|---|---|---|
| | 12 | 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( ).<br>  <br> |
| 10 | 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?<br>يمكن تحويل كائن من نوع Equestrian إلى نوع Racer بشكل مباشر ؟<br>السبب:<br>كلا الفئتين Equestrinan , Racer. ترثان من فئة Horse<br>ولاتوجد علاقة وراثه مباشره بينهما<br>. |

| | | |
|---|---|---|
| | | |
| 11 | 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!”<br><br>pu public class ArrayOutOfBoundsExample {<br>public static void main(String[] args) {<br>int[] numbers = {10, 20, 30};<br>try {<br>int value = numbers[3];<br>System.out.println("قيمة العنصر: " + value);<br>} catch (ArrayIndexOutOfBoundsException e) {<br>System.out.println("حدث خطأ");<br>}<br>} |
| 12 | 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.<br><br>public void makePayment(double amount) { *// make a payment*<br>if(amount<0)<br>throw new IllegalArgumentException("Negative Amount is not Allowed");<br>balance -= amount;<br>} |

```
public void makePayment(double amount) {
    if (amount < 0) {
        throw new IllegalArgumentException("Negative Amount is not
Allowed");
    }
    balance -= amount;
}
```