CST Part IA: OOP, SV 4
Joe Yan
2016-12-19

1 2009P1Q8

(a) i. The most positive value of byte is 127.
The most negative value of byte is -128.

ii.

$$sum = \begin{cases} 0 & -2^{31} \leqslant n \leqslant 0 \\ \sum_{i=0}^{n-1} 2^i & 1 \leqslant n \leqslant 7 \\ -1 & 8 \leqslant n \leqslant 2^{31}-1 \end{cases}$$

I suppose $1 \leqslant n \leqslant 7$ is what the examiner looking for here.

i. protected TreeMap$< String, Topic >$ topics
protected TreeMap$< Date, Message >$ messages
Or protected ArrayList$< Message >$ messages is fine if the information of creation date is not required. (Keeping add the new message to the tail of the List can automatically sort the list by creation time and so date.)

ii.
```
public void addTopics(String name){
                topics.put(name, new Topic());
}
public void displayMessages(){
        foreach(Message mes : messages){
                mes.display();
        }
}
```

iii. An Observer Pattern is a proper choice here. For the UML diagram, the modification will be: Topic has * User.

Topic
... # users

...
+subscribeUser(User u)
+unsubscribeUser(User u)
+notifyByEmail(Message m)

User
...

...
+subscribeTopic(Topic t)
+unsubscribeTopic(Topic t)

The users can be a LinkedList. (Elements can be deleted at any position.)
subscribeUser is a method add a new User to the Collection users.
subscribeTopic just calls the subscribeUser method for convenience.
un.. methods are similar.
notifyByEmail call email method for each User in Collection user and can be called in addMessage at the end to make sure when a new message is added, it will be emailed to all users.

2 1998P3Q3

(a) Java provided classes and interfaces for performing the OOP concepts. Classes can be defined as abstract (unable to be instantiated and normally has abstract methods waiting for implementation) or final (unable to be extended). Extending a existed class will enable us to add more features to the origin class if the extended class is abstract and the child class is not the unfinished signature must be implemented. The interface is a purely abstract class with no states and all methods abstract. Java does not allow multiple inheritance directly (one class can only extend at most one other class) but it can be similarly performed by implementing multiple interfaces and this also avoids method inheritance collision. (at most one non-abstract method up to the inheritance tree)
Java create a new object when *new xxxObject*() is called. It first check whether this the first ever object of the class. If it is not The Java will load the *xxxObject.class*, create *java.lang.class* object, allocate static fields in the memory and run the static initialiser blocks (static...). If there is already some objects of this class Java will create a reference (address in the memory and type) in the stack and allocate the memory in heap for the object, run non-static initialiser blocks (...) and run the constructor to initialise the created objects.

(b) Abstract class can not be instantiated and may contain some or no abstract methods.
Interface is a kind of special classes which is unable to be instantiated with only abstract methods and no fields.
Both of them are designed for the OOP concept of Java. Abstract class prevented the user to created a object of the class and force them to extend the abstract class to implement the abstract method in different way. Interface make the idea of multi-inheritance possible and avoided the method inheritance collision.(in (a))

(c) (public,package,protected,private) makes the identifier visible for (everyone, the same package, the same package and subclass, the same class).

(d) We have Swing now.

3 1998P1Q3

(a) >>> operator will remove the least significant bit and move other bits to this removed bit and fill the empty most significant bit with a 0.
>> operator will do the same as ¿¿¿ but it will not change the sign of the value input which should be decided by the most significant bit.

(b) Java decided to take $0.0d/0.0$ as a special value for double ($0.0f/0.0$ the same for float). This is defined Double.NaN but this is a range of value and the actually bits stored is unsure so the == compare the primitive type value for
$double\ a = Double.NaN;$
$a == a;$
It returns false (the actual compared bits in the memory is unsure).

(c) final keyword will set the variable constant (unchangeable after initialisation) or deny all request to override the final method or extend the final class.
finally keyword is in the syntax try...catch(...)...finally.... The code in the finally block will be called after any exception is caught and it even cannot be skipped by return!

(d) In default, Java will return "three3" for this expression. Java casts the 3 to "3" and the + operator for two String means concat so we get "three3".

(e) I suppose this piece of code wants to create an Array of int of length 10 and store 10 1's. Fixed code:

```
...
        int[] a = new int[10]; \\syntax error
        for (int i=0; i<=9; ++i) \\subfix should be from 0
                a[i] = 1-a[i];
...
```