Print

Tips N Tricks - SCJP 5.0 Tiger Exam

1. About 30% of the questions in the SCJP 5.0 exam are Drag and Drop questions. The answer gets reset if you try to review it. So, always complete and double-check your answer before moving away from a Drag and Drop question.

2. It would be helpful to solve Drag and Drop questions on a sheet of paper before actually attempting them in the test.

3. Just reading about the concepts and seeing examples will not make things clear. Make a small example of your own to test every new concept/rule you learn. This will help you understand it better. Also, alter the example and experiment with different possibilities. You will learn and retain much better if you follow this process.

   Let us say, you read about the rule that static methods cannot be overridden. It is not very clear what will happen if you attempt to override a static method, whether this will cause a compiler error or not.

   To explore this, you can make an example similar to this one.

```
class Base{

static void f() { System.out.println("base"); }}

class Derived extends Base{

static void f() { System.out.println("derived"); }

public static void main(String args[]){

Base base=new Derived();

base.f();

}

}
```

   Now try to compile, there are no errors. So, it means that it is not illegal to have the same static method in the base and derived class. But when you invoke the method, the actual method being invoked is the one declared in the base class. This is because the method invoked depends on the declared type of the reference variable. That is why you say that overriding does not happen in this case. If this had been a non-static method, the method invoked would have been the one declared in the subclass. Now things are clear to you.

   Once this concept is clear, you can start trying different possibilities, such as

   - What happens if you override a static method with a non-static one or vice versa?
   - What will happen if you try to declare a static method abstract?
   - What will happen if you declare a static method to be synchronized?

   You can think of many options like this. Make small variations to the program and try out all of them. Instead of learning just one concept, you will get a much deeper understanding of many related concepts.

4. The revised exam features more performance-based questions compared to knowledge-based questions which require memorization of API details. So spend more time on hands-on practice, especially of commonly used methods, and avoid trying to memorize each and every API detail simply by reading.

5. A question might not pertain to a single objective, but may often test your expertise in various topics. For example, a question from the topic 'Collections/Generics' might use autoboxing and the enhanced "for?loop. So, while

you practice one topic, try to associate it with other topics as well.

6. Be clear about the classes implementing the Collection types like List, Set, and Map. Be aware of which of these types support sorting, which allow duplicates, which give the best performance, and so on.

7. When you see the wait() and notify() methods, first ensure that they are called within a synchronized context. Also remember that they are defined in the Object class, so they are invoked on the object which is locked and not on the thread itself.

8. Remember that the scope of a variable declared within simple for, enhanced for, while, or do-while loops is restricted to the loop itself.

9. Read the choices properly; don't get confused between “import?statements and “import static?statements.

10. Whenever you see classes implementing interfaces, always check if the methods have been defined as public. Also, check if all the methods from the interface have been defined in the class. If this is not the case, make sure the class is declared abstract.

11. Remember that immutable string objects are given by String class. StringBuilder and StringBuffer both give mutable strings; the only difference between them is that StringBuffer is thread-safe, while the other is not.

12. When you see the usage of varargs in a method, ensure that the ellipsis is appearing as the last argument to the method.

13. Remember that the super type of all collections is Collection<?> and not Collection<Object>.

14. Remember that since the wild card (?) stands for an unknown type, you cannot add elements to generic types using wild cards.

15. "Inner classes" is one of the most confusing sections. You need to memorize the correct syntax, the different ways of instantiating them, and their access rules. Even questions from other objectives might be using inner classes. So try to practice a lot in this area.

    Anonymous inner classes are especially prone to errors.

    Watch out for small syntax errors like the missing semicolon at the end of this anonymous inner class defined inside a method argument.

16. Know the distinction between Comparable and Comparator and the methods defined by these interfaces.

17. A subclass can overload the methods defined in both itself and its superclass, don't confuse this with overriding.

18. Use the following simple rules to distinguish between overriding and overloading.

    - In overriding, the arguments and return type are the same. The exception is ‘Covariant return types' which allow a method in a subclass to return an object whose type is a subclass of the type returned by the method with the same signature in the superclass.
    - In overloading, the arguments must be different and the return type does not matter.
    - If the return types differ (cases not satisfying covariant return types), but arguments match, the code is not legal. Means the code will not compile.

19. Threads and Garbage collection are two topics in Java which have some platform/implementation-dependent features. You should be able to clearly differentiate between such behavior and features that actually follow the rules and hence, are predictable. For example, garbage collection algorithm varies across different implementations. Garbage collection cannot be forced to happen, you can only request for it. So there are no guarantees here. Now let us see what you can be sure about. The finalize() method of an object will be surely called once before it is garbage collected. This rule will not be violated, whatever the implementation is. But again, you need to remember that finalize() might never be invoked on the

object because the garbage collector might never run during the program execution.

20. Don't confuse between the finalize method and the finally block. They have nothing to do with each other. Any code defined within the finally block will always be called whether an exception happens or not. The only exception is a System.exit call before the control reaches the finally block.

21. If the subclass does not define any constructors, and is instantiated, then its superclass must have a no-argument constructor since an implicit super() call will be made.

22. Remember that sleep() and yield() are static methods of the Thread class, and are invoked on the currently executing thread.

23. Remember that the order in which the threads run is decided by the thread scheduler, though we can request to change the priority. Also, calling the start() method only puts the threads in the runnable state, the run() method gets called only when the thread gets the CPU. Note that calling the run() method directly will not spawn a new thread.

24. Be clear about the difference between the legal and appropriate uses of assertions. If the assertion syntax is correct then the code is legal. But there are some situations where the use of assertions is inappropriate, even though it is legal. For example, assertion expressions must neither alter the program state nor have any side effects that would not be perceived if assertions were turned off.

25. The objective on Garbage collection will have questions asking how many objects are eligible for garbage collection at a particular point in the program. The answers to such questions cannot be verified with the help of examples because you cannot force the garbage collector to run. So attempt as many questions as possible on this topic from various mock exams, compare your answers to test your knowledge.

26. Though it is difficult to use the JLS (Java Language Specifications) as a tutorial to learn concepts, it should be used as an authoritative reference to check if you have understood the rules clearly.

27. Know the difference between using ++ as a postfix and prefix operator. Also, you must be aware of why && and || are called short-circuit operators and how they differ from & and |.

28. Ideally, you must have at least six months experience before you go for the SCJP Tiger exam. So get a lot of hands-on practice, especially on the new features introduced in Java 5.0, like Generics.

29. Once you have gone through all the concepts, start taking as many mock exams as possible to test your knowledge. After taking each mock exam, note down the mistakes you made and work upon them. On the day before the exam, you can go through them once again and ensure that you have mastered your weak areas.

30. Try to be relaxed and sleep well on the night before the exam.

31. Bring some water in the exam room and drink regularly in order to maintain a fair water balance in your brain.