Points To Remember for SCJP
www.techfaq360.com

The datatype in a switch statement must be convertible to int, i.e., only byte, short, char and int can be used in a switch statement, and the range of the datatype used must cover all of the cases (case statements).

An integer literal (w/o a decimal pt) is assumed to be a 32-bit int primitive, but a value containing a decimal point is assumed to be a 64-bit double.

Wrapper classes for primitives produce immutable objects.

• Java characters are 16-bit unicode characters, and they can be initialized in the following ways:

```
char c1 = '\u0057'
char c1 = 'w'
char c1 = 87
char c1 = (char) 87          (acc to RHE)
char c1 = '\r'
```

• instanceof operator can take a class, an interface or an array type as an argument.

• To test if an object is an array, use: myObject.getClass().isArray();

• MenuContainer interface is implemented by Container, Frame, Menu and MenuBar

• An identifier must begin with a letter, a dollar sign ($), or an underscore. Subsequent characters maybe letters, $, underscore, or digits, but NOT other characters liek %, &, etc.

• The keywords syncrhonized, private and protected cannot be used with a class.

• A variable declared inside a try block cannot be used in the catch block.

• If no super-class constructor is explicitly called, and there is no call to other constructor (using this(..)), the default constructor of the superclass (w/o any arguments) is automatically called.

• wait(), notify() and notifyAll() are the instance methods of Object class, not Thread class (but they are inherited by Thread class).

• stop(), suspend() and resume() are deprecated methods in Java 1.2

• References to member methods are resolved at runtime using the type of object, but the references to member variables are computed at compile time using the type of the reference.

• Javadoc documentation does not list private variables and methods.

• A valid overload differs in teh number or types of arguments. Difference is return type only is not a valid overload.

• Inner classes defined local to a block may not be declared to be static.

• Mmember inner classes cannot have the same name as enclosing class, but they CAN subclass the enclosing class.

• If an instance of the inner class had to refer to the associated instance of the enclosing class, then the following syntax could be used:

```
EnclosingClass myPart = EnclosingClass.this;
```

• No datatype is automatically converted to a char. However, you can cast to a char.

• Solaris uses pre-emptive scheduling; Windows & Mac uses time-slicing.

• A monitor is any object that has some synchronized code.

• If you have a sycnhronized method in a class, a subclass with an overriding method does not

have to declare it to be synchronized.
• A static method may not be overridden to be non-static, and vice-versa.
• equals() method just returns false even if the object reference types are different. It does not throw an exception.
• The add(..) method of the Set returns false if you atttempt to add an element with a duplicate value, but no exception is thrown.
• Constructors CAN be private.
• Anonymous classes cannot have constructors.
• -(i) = ~(i) + 1
• The test condition (assuming i is an int):
If (i=2)
will give a compiler error because the test condition expects a boolean and the assignment (i=2) returns an int
• While using the RandomAccessFile constructor, if the file is not present, a mode "r" throws a FileNotFoundException. If "rw" mode is used, a new file is created with zero length.
• Applets can't have a constructor with arguments. It can have constructor with no arguments.
• Any component can have menu barsor pull-down menus, although only frames have a method for adding a menubar.
• Double.NaN == Double.NaN //returns false
But, if Y = X = Double.NaN, X.equals (Y) returns true
• Also, +0.0 == -0.0 //returns true
But, if Y = +0.0 and X = -0.0 , X.equals (Y) returns false
• For shift operators, >>, << and >>>, unary numeric promotion is applied to each operand separately, and the datatype of the resulting expression is the same as the left operand.
• instanceof is not same as instanceOf.
The instanceof operator tests whether its first operand is an instance of its second.
op1 instanceof op2
op1 must be the name of an object and op2 must be the name of a class.
An object is considered to be an instance of a class if that object directly or indirectly descends from that class.
There is nothing like instanceOf in java
x instanceof P If x is null, the instanceof test simply returns FALSE --it doesn't cause an exception
String[] str = new String[10]; now
str instanceof String[] ===returns true
str instanceof Object ===returns true
str instanceof Object[] ===returns true
str instanceof String ===Compilation Error
• If a = null, System.out.println (a); prints null, and does not throw an exception.
• A class SHOULD (MUST) be declared abstract if it has one or more abstract methods. An abstract class can also have non-abstract methods. You can also declare a class to be abstract even when it has no abstract methods, so that users can't instantiate it and are forced to subclass

it.
• In an interface, ALL methods must be abstract. Infact, all methods in an interface are implicitly abstract. Also, all methods and variables in an interface are implicitly public, and must declare them to be public while implementing the interface.
• It is a compile-time error for a constructor to invoke itself by calling this(), but there is a bug which allows cyclic constructor invocation, that is constructor 1 calls constructor 2 and vice-versa.
• If you call Thread.sleep() in a synchronized code, the thread does not give up the lock, it keeps the lock during sleeping. It gives up the lock only when calling the method wait().
• abs (Integer.MIN_VALUE) = Integer.MIN_VALUE
• No duplicate objects are allowed in a Set, and duplicate means that the equals() method returns true. So, there can't be two separate objects having same value in a set.
• If run() is a static method, and me() is also a static method returning null, then me().run() WILL compile and run correctly. In other words,

((StaticTest)(null)).run(); //will work fine (where StaticTest is a class)

• Byte b1 = new Byte("172")
b1.toString() == b1.toString() is false.
The toString() method for the Wrapper classes (except Boolean) creates a string over the heap, and returns a reference to that string. A new string is created eacht time, hence the result false.
• Each instance of a thread belongs in a ThreadGroup (even if you don't explicitly set it). In the latter case, the threadgroup is the same that of the thread which created it.
• If i=0
i++ will result in i=1
i=i++ will not change the value of i. It'll remain at 0.
• You can use super.var to access a variable in the superclass.
• Even new Boolean (null); will create a Boolean with a false value, and won't throw an exception. This is true although null is a null literal, and not the same as "null" string.
• Interfaces can have access modifiers of public or blank, just like classes.
• UTF-8 is ASCII
• An array of primitives of on type canot be cast into an array of primitives of another type.
• The \uxxxx notation can be used anywhere in the source to represent Unicode characters, e.g.,
char a = '\u0061'
ch\u0061r a = 'a
char \u0061 = 'a'
are all valid (and equivalent) statements.
• An empty file is a valid source file. A source file can contain an optional package declaration, any number of import statements and any number of classes and interface definitions.
• The modulus operator (%) can be used with integral as well as floating datatypes.
• Extended assignment operators, e.g., *=, += ensure than an implicit narrowing conversion takes place which makes the result fit into the target variable, e.g.,
b = b + i; // is illegal.
b+=i //is legal
• In an assignment of the following type , where i=0:

a[i] = i = 9;
a[0] will be assigned the value of i=9, i.e., 9, as opposed to a[9], which you might think. That is, the initial value is used to determine which element is to be assigned the value.
• Elements in an unitialized array object get the default value corresponding to the type of elements. This is true, regardless of whether the variable is an instance variable or a local variable.
• Local variables cannot be declared to be static and cannot be given an accessibility modifier.
• The declaration of a non-abstract method must provide an implementation ({ }, at least !!).
• A constructor does not declare any return type, not even a void.
• A constructor cannot be final, abstract or static.
• Normal methods (non-constructor) MUST specify a return type, at least a void !!
• You cannot pass void as a parameter to a method, e.g., void method (void) is wrong.
• Anonymous arrays, (analogous to anonymous classes) have a form:
new int[] {3,2,4,6,1}
and are usually used for being passed to a method as a parameter.
• The main() method CAN throw checked exceptions.
• It is not possible to use a break statement inside an if block. Break statements can only be used for
o Do-while
o While
o For
o Labelled blocks
o Switch statements
• catch and finally blocks can themselves throw checked exceptions which are handled in the usual way.
• An overriding method must have the same method name, same parameters, and same return type as the original method.
• Whether parameters in the overriding method should be declared final is at the discretion of the subclass.
• Only methods that are accessible may be overridden. So, private methods can't be overridden.
• this() and super() cannot both occur in the same constructor.
• If a constructor does not have either a this() or a super() as its first statement, then a super() call to the default constructor of the superclass is automatically inserted.
• An interface can define constants. Such constants are considered to be public, static and final regardless of whether these modifiers are specified or not. In the case of multiple inheritance, any name conflicts are resolved using fully qualified names.
• A non-static inner class cannot have any static members.
• Package member classes, local classes and anonymous classes cannot be declared to be static.
• The static initializer block cannot be contained in a method.
• In a method declaration, the modifiers, e.g., public, static, must precede the return type.
• All method defined in Set are also defined in Collection, but the same is not true for List.
• Java returns the same string object reference for almost all of the methods invoked on a string

object, provided the result of the operation happened to be the same as that of the string object on which the method is invoked.
• According to the preceding point, a comparison like this:
"abc".substring(0)=="abc" would return true.
• For the wrapper classes, a new string reference is created each time, except for Boolean class, which returns a reference to "true" or "false" from the string pool.
• String comparisons for strings created at runtime, return false, e.g., if ja="ja" and va = "va" and java="java"
java == "java" //is true
java = "ja" + "va" //is true
java = "ja" + va //is false
• A vector can only store object references, NOT primitives.
• length is a field (instance variable) for an array, but a string length() is a method.
• Math.round (-4.7) = -5
• Math.ceil (-4.7) = -4.0
• Math.floor (-4.7) = -5.0
• Native methods cannot be abstract, but they can be static.
• Garbage collection cannot be turned off.
• double c = Math.floor (Double.MIN_VALUE) // = 0.0
• double c = Math.ceil (Double.MIN_VALUE) // = 1.0
• For Math.round()
o Add 0.5 to the number to be rounded.
o Do a Math.floor() on that number.
• getValue() method of the Adjustment class returns an int value for the adjustment setting.
• When a thread executes a waitForID() call on a MediaTracker, the current thread stops executing.
• The finally block is always executed except when there is a System.exit call.
• Local inner classes cannot be declared public, private, protected, or static.
• Member inner classes can be private, protected, public, final or abstract.
• Component class is an abstract class.
• Vector v = new Vector (initial capacity, capacity increment), e.g.,
Vector v = new Vector (5, 10), where 5 is the initial capacity of the vector and 10 is the capacity increment.
• There is no concat() method for StringBuffer class.
• replace() is a String method.
• The read() method returns -1 when it reaches end of the file.
• Using the modifier static or final with transient is legal.
• An abstract method cannot be private, static, final, native or synchronized.
• Object method equals() will throw a NullPointerException if the calling object is null. If the argument is null, it returns false.
• If a method is there in a subclass, but not in the base class, then you can't use a reference variable of the base type to access that method, even if the object is of the subclass type. You'll be

required to cast the reference to subclass type.
• getWhen() method of InputEvent class is used to extract the instant at which the InputEvent (e.g., MouseEvent) was generated.
• Boolean and Character classes do no extend java.lang.Number.
• Character class does not have a valueOf() mehod.
• add() and addItem() both can be used to add an item to a choice.
• Math.ceil (-0.5) = -0.0 (not 0.0 !!)
• The order of the floating/double values is:
-infinity -> -ve nos/fractions -> -0.0 -> 0.0 -> +0.0 -> +ve nos/fractions -> +infinity
• 2 + 3 + "" = 5
• 2 + "" + 3 = 23
• " + 2 + 3 = 23
• A label CAN be placed in front of another label.
• You can't assign a 2nd value to a final variable. You CAN assign the value like this:
final int MAX_INT;
MAX_INT = 30;
but you cannot reassign the value.
• null, true and false, are NOT keywords. They are literals. null is a NULL literal. True and false are boolean literals. But remember that they ARE reserved words.
• You can put a semicolon after '}'
• If i = 3, then
print3 (i, ++i, ++i +++i) is equivalent to print3 (3, 4, 5+6)
print3 (i, ++i, i+++ i ++) is equivalent to print3 (3, 4, 4+5)
• Once an array is created, its length can never be changed.
• The array-size specifier must be an integral type, you can't use long, etc.
• StringBuffer class does not ovverride equals method. So, the equals method returns false when passed two different StringBuffer objects, containing the same values.
• float f = 40.0 // illegal because 40.0 is double
• byte b = 40 // legal although 40 is int
• int arr[] = new int [-10]
compiles correctly, but throws a NegativeArraySizeException at runtime.
• Static variables CAN exist inside an inner class if the inner class is also static.
• square() or sqr() methods do not exist for Math class, because same functionality can be provided by pow().
• An inner class can actually be a subclass of the outer class.
• Runtime exceptions can be avoided with a correctly coded program. However, checked exceptions can arise even in a correctly coded program.
• The implementation of toString() method for the String class simply returns a reference to the current object, i.e.,
String s1 = new String("Java");
String s2 = s1.toString();
s1 == s2 // is true

• abs, min, max and round are the only methods (out of the methods which you need to remember for SCJP ;-)) of the Math class with return an int (also).
• There is a static method for Integer class:
public static String toString(int I)
which can be used to return a String object representing the specified integer. There are similar methods for Long, Float, Double, Byte, Short.
• Overriding methods cannot throw checked exceptions not thrown by overridden methods, but they CAN throw unchecked exceptions not thrown by overridden methods, like ArrayIndexOutofBoundsException.
• Unary numeric promotion is not applied to ++ and –
• You can also run a class from the command line, even if it is not declared public.
• protected means accessible by all the subclass, as well as all the classes in the same package.
• When -ve signs are involved in % operator, compute the result by ignoring the sign(s) and then, just apply the sign of the left operand to the result, e.g.,
-7 % -2 = - (7%2) = - 1
• instanceof operator simply returns false if the left hand argument is a null value. It does not throw an exception.
• The default no-argument constructor created by Java is public.
• When asked to write a statement, be sure to end it with a semicolon.
• The order of the exceptions specified by a method should be more specific exception followed by less specific exception.
• A try block must be followed by one of these combinations:
o One or more catch blocks.
o A finally block.
o Both catch and finally blocks.
• The main method can be declred without the public modifier, and it'll run fine.
• A more specific catch block cannot follow a general one. The program won't compile.
• If you have two methods in a class:
o public int method (int b, float x)
o public int method (float b, int x)
and you call one of these methods as method (2, 5), the program won't compile, because the reference to the method is ambigous. However, if you call the method as method (2, 5.0), it'll compile and run fine.
• reverse() is a method of StringBuffer class, not String class.
• In case of shift-operators, only primitive integral types (byte, short, char, int, and long) can be used, and unary numeric promotion is applied separately to each operand.