TIP: If you are curious as to what Unicode characters are “letters” as far as Java is concerned, you can use the isJavaIdentifierStart and isJavaIdentifierPart methods in th**e Character** class to check.

TIP: Even though $ is a valid Java letter, you should not use it in your own code. It is intended for names that are generated by the Java compiler and other tools. Any currency symbol could be used as identifier.

Note that integer division by 0 raises an exception, whereas floating-point division by 0 yields an infinite or NaN result.

double y = Math.sqrt(x);

System.out.println(y); // prints 2.0

NOTE: There is a subtle difference between the println method and the sqrt method.The println method operates on the System.out object. But the sqrt method in the Math class does not operate on any object. Such a method is called a static method.

When a number is present in the code, it is called a literal.

A local variable is a variable defined within a method. Local variables must be initialized before use.

Think of the acronym PIC (picture): package, import, and class. Fields and methods are easier to remember because they merely must be inside of a class.

**Objects vs. References**

Do not confuse a reference with the object that it refers to; they are two different entities. The reference is a variable that has a name and can be used to access the contents of an object. A reference can be assigned to another reference, passed to a method, or returned from a method. **All references are the same size**, no matter what their type is. An object sits on the heap and does not have a name. Therefore, you have no way to access an object except through a reference. Objects come in all different shapes and sizes and consume varying amounts of memory. An object cannot be assigned to another

**finalize()**

Java allows objects to implement a method called finalize()that might get called. This method gets called if the GC tries to collect the object. If the GS doesn’t run, the method doesn’t get called. If the GS fails to collect the object and tries to run it again later, the method doesn’t get called a second time.

Simple Java was intended to be simpler than C++. In addition to eliminating pointers, it got rid of operator overloading. In C++, you could write a + b and have it mean almost anything. But Java still has an overloaded operator +, which could be used either for numbers or for concatenation.

Understand the effect of using packages and imports. Packages contain Java classes. Classes can be imported by class name or wildcard. Wildcards do not look at subdirectories. In the event of a import conflict, class name imports take precedence over wildcard \*.

**Overflow vs Underflow**

Overflow is when the absolute value of the number is too high for the computer to represent it.

Underflow is when the absolute value of the number is too close to zero for the computer to represent it.

You can get overflow with both integers and floating-point numbers. You can only get underflow with floating point numbers.

To get an overflow, repeatedly multiply a number by ten. To get an underflow repeatedly divide it by ten.

If the variable x is a signed byte it can have values in the range -128 to +127, then

x = 127

x = x + 1

will result in an overflow. +128 is not a valid value for x.

For floating point numbers, the range depends on their representation. If x is a single precision (32-bit IEEE) number, then

x = 1e-38

x = x / 1000

will result in an underflow. 1e-42 is not a valid value for x.

int x = 2, z = 3;

x = x \* z; // Simple assignment operator

x \*= z; // Compound assignment operator

The left-hand side of the compound operator can only be applied to a variable that is already defined and cannot be used to declare a new variable. In the previous example, if x was not already defined, then the expression x \*= z would not compile.

Exclusive OR (^) is only true if the operands are different.

**Comparing for and for-each Loops**

Since for and for-each both use the same keyword, you might be wondering how they are related. While this discussion is out of scope for the exam, let’s take a moment to explore how for-each loops are converted to for loops by the compiler. When for-each was introduced in Java 5, it was added as a compile-time enhancement. This means that Java converts the for-each loop into a standard for loop during compilation. For example, assuming names is an array of String[] as we saw in the ﬁrst example, the following two loops are equivalent:

for(String name : names) {

System.out.print(name + ", ");

}

for(int i=0; i < names.length; i++) {

String name = names[i];

System.out.print(name + ", ");

}

For objects that inherit java.lang.Iterable, there is a different, but similar, conversion. For example, assuming values is an instance of List<Integer>, as we saw in the second example, the following two loops are equivalent:

for(int value : values) {

System.out.print(value + ", ");

}

for(java.util.Iterator<Integer> i = values.iterator(); i.hasNext(); ) {

int value = i.next();

System.out.print(value + ", ");

}

Notice that in the second version, there is no update statement as it is not required when

using the java.util.Iterator class.

An object is a runtime instance of a class in memory. All the various objects of all the different classes represent the state of your program.

Java classes have two primary elements: **methods**, often called functions or procedures in other languages, and **fields**, more generally known as variables. Together these are called the **members** of the class. Variables hold the state of the program, and methods operate on that state. If the change is important to remember, a variable store that change.

Java building blocks:

* class
* interface
* enum
* annotation

Java calls a word with special meaning a keyword

Method requires information be supplied to it from the calling method, this information is called a parameter. But concrete parameter value is an argument

Method signature – method name and parameters (parameters order plays a role, namely if parameters types are different than different order of them result in different signature)

Comments aren’t executable code and could be place anywhere

Spaces between / and \* (or \*\*) in multiline comment cause compile error

**Classes vs. Files**

* It is possible to put several classes in single \*.java file (not common practice, but anyway).
* And such files aren't mandatory to be a public
* To compile them use javac with common java file
* To run use java with class name (it means that param for javac is class name, not class-file name)
* It is possible to have at most one public class in \*.java file. And in such a case class name and java file name should be the same. But public class couldn’t be presented at all.

Bytecode consists of (~256 that is why byte) instructions that the JVM knows how to execute. Notice that we must omit the .class extension to run Zoo.java because the period has a reserved meaning in the JVM.

The keyword static binds a method to its class, so it can be called by just the class name. Java doesn’t need to create an object to call the main() method, in fact, the JVM does this when loading the class name given to it.

If a main() method isn’t present in the class we name with the .java executable, the process will throw an error and terminate. Even if a main() method is present, Java will throw an exception if it isn’t static. A non-static main() method might as well be invisible from the point of view of the JVM.

Java executables – classes which run using ‘java className’

The keyword **void** represents the return type. A method that returns no data returns control to the caller silently. In general, it’s good practice to use void for methods that change an object’s state. In that sense, the main() method changes the program state from started to finished

main() method’s parameter list, represented as an array of java.lang.String objects. In practice, you can write String[] args, String args[] or String... args; the compiler accepts any of these. The variable name args hints that this list contains values that were read in (arguments) when the JVM started. You can use any name you like, though.

The characters ... are called varargs (variable argument lists)

Spaces are used to separate the arguments for java command. If you want spaces inside an argument, you need to use quotes.