• Identify correctly constructed source files, package declarations, import statements, class declarations (of all forms including inner classes), interface declarations and implementations (for java.lang.Runnable or other interface described in the test), method declarations (including the main method that is used to start execution of a class), variable declarations and identifiers.

Structure of source files:
• An empty file is a valid java src file.
• Package statement (if exists) should be the first statement. (comments before it are ok!). Next should be the import statements. (if any).
• Last, should be the class/interface declaration.
• Method declaration for the standard main:
public static void main(String[] args);
• Main method can also be final, native, synchronized. No matter whether other declarations (like private, protected) work on your m/c, for the purpose of the exam, it should be public.

Variable Declarations and Identifiers:
Rules:
• A valid java identifier is composed of a sequence of java letters (this includes uppercase and lowercase ASCII latin letters and _ , $) and digits, the first of which must be a letter. (Valid: _123, a$2 NOT VALID: 123$, goto)
• It cannot be same as any java Keywords (eg. while, for, class etc) or literals (ie. true, false or null).
• Class names can serve as a valid identifier. eg. String String = "asd"; //This is valid.
• Pitfalls:
• If you have a main method like: public static int main(String[] args){ return 10; } (Note return type.) It will compile but the program will throw exception at runtime saying there is no main method. Same will be the case if you have: public static void main(String args){}
• A class without a main method may be run by JVM, if its base class has a valid main method.

• State the correspondence between index values in the argument array passed to a main method and command line arguments. Identify all Java Programming Language keywords and correctly constructed identifiers.

Consider this:

```
public class TestClass
{
  public static void main(String[] args)
  {
    System.out.println("Hello, World!");
```

```
  }
}
```
Points to remember:
• args will NEVER be null.
• If no argument is passed, args.length will be 0.
• If the above program is run with the command line: "java TestClass hello world", then args[0] will be "hello" and args[1] will be "world".
• UNLIKE IN C/C++, the word 'java' or the name of the class is not passed.
• Language keywords: Here's a list of Java's keywords. These words are reserved--you cannot use any of these words as names in your Java programs. true, false, and null are not keywords but they are reserved words, so you cannot use them as names in your programs either.

• State the effect of using a variable or array element of any kind when no explicit assignment has been made to it.

Three Important points:
• Class members (static/non-static) are ALWAYS initialized automatically to their default values.
• Variables declared in methods (local variable) are NEVER initialized automatically. You must initialize them before using them.
• Whenever you "ALLOCATE" an array (ie. new int[3] or new Object[5] etc), the elements are automatically initialized to the default value of their type. Understand what is meant by default values: It is the value assigned by the JVM to a variable.
Primitive variables are initialized to 0 (for integral types: byte, char, short, int, long), 0.0 (float and double), false (booleans).
Object variables (including arrays) are initialized to null.
Example:

```
class TestClass
{
  int   i; //initialized to 0
  float f; //initialized to 0.0
  boolean bool; //initialized to false

  int[] iA;     //initialized to null: S.o.p(iA) will print 'null'
                //S.o.p(iA[0]) will throw NullPointerException

  String[] sA;  //initialized to null: S.o.p(sA) will print 'null'


  public void m1() //to see about local variables
  {
    int k; //WILL NOT BE INITIALIZED. S.o.p (k); ERROR, k is not initialized!
```

```
    int[] jA; //WILL NOT BE INITIALIZED. S.o.p (jA); ERROR, jA is not initialized!

    int[] kA = new int[3]; //elements are initialized automatically to { 0, 0, 0} because default value of int is 0

    String[] sA = new String[3]; //elements are initialized automatically to { null, null, null} because default value of Object is null.

  }

}
```

• State the range of all primitive data types and declare literal values for String and all primitive types using all permitted formats, bases, and representations.

Range of primitives:
boolean : true/false
byte (8 bits => 2^8 values) : -128 to 127 (-2^7 to 2^7 -1)
short (16 bits => 2^16 values) : -32768 to 32767 (-2^15 to 2^15 -1)
char (16 bits => 2^16 values) : 0 to 65536 (0 to 2^16 -1)
int (32 bits => 2^32 values) :   -2^31 to 2^31 -1
long (64 bits => 2^64 values) : -2^63 to 2^63 -1
float 32 bits
double 32 bits
.3e2 is a valid float but e2 is not. e2 is parsed as a variable name.
Octal numbers are written by prepending 0 in front of the no. Eg. 012 is 12 in octal.
Hex numbers are written by prepending 0x in front of the no. Eg. 0x12 is 12 in hex.