JAVA BASICS

VARIABLES

1) Variable names must always begin with a lower case letter and must always be declared prior to their use.
2) Examples of variable declarations:
   a) int numberOfItems;
   b) String msg;
   c) Television tv1;

CLASSES

1) Class names must always start with an upper-case letter
2) A class defines an abstract data type: a set of values and a set of operations on those values.
3) A class is a collection of values (attributes/variables) and a set of operations (methods). The ordering of the elements in the class doesn’t matter (i.e. the location of the methods and attributes can be moved around without changing anything. Conventionally, variables come before methods but this is not required).
4) Here is an example of a class that defines three operations, five attributes, and one constructor.

```java
public class Student {
    private String first, last, id;
    private double gradePointAverage;
    private boolean transferStudent;
    public Student(String f, String l, String id, double gpa, boolean trans) {
        first = f; last = l; this.id = id; gradePointAverage = gpa; transferStudent = trans;
    }
    public boolean isTransferStudent() { return transferStudent; }
    public String getInitials() { return first.charAt(0) + last.charAt(0); }
    public boolean qualifiesForScholarship(double reqGPA) { return gradePointAverage >= reqGPA; }
}
```

ARRAYS

1) Arrays are indicated by square brackets
2) Array variables, just like all others, must be declared before being used.
3) Array variables must be created before being used.
4) Array elements must be created and placed into the array before the array has any real usefulness.
5) Assume that ‘x’ is an array. The only expressions that are meaningful with respect to ‘x’ are: x.length and x[i] where i is an int in the range 0 to x.length-1. Note that the following are not valid expressions: x.reset(), x.next(), x.hasNext(), x.get(0), x.insert(y).
6) Examples of using arrays:
   a) int[] values; // declare a variable named ‘values’ to be of type ‘array of ints’
   b) values = new int[10]; // create an object of type ‘array of ints’ that has ten slots
   c) for(int i=0; i<values.length; i++) { values[i] = i*2; } //place numbers into the ten slots of the array named ‘values’
**STRINGS**

1) A String is not an array
2) A String is not a list
3) A String is not a char
4) A String object cannot ever be changed.
5) Assume that ‘x’ is a String. There are many expressions that are meaningful with respect to ‘x’. You should scan the Java API for a complete list. The most common expressions are: x.length(), x.charAt(i), x.indexOf(c), x.contains(s), x.startsWith(s), x.substring(i,j) where variables i, j, c and s are of the correct type for their functional use.

6) Examples:
   a) String s = “UWL rocks”;
   b) String t = s.substring(0,4); // t is now the String value “UWL”
   c) char c = s.charAt(2); // c is now the char value ‘L’
   d) int i = s.indexOf(’r’); // i is now the int value 4
   e) boolean f = s.contains(“rock”); // f is the boolean value true
   f) boolean g = s.startsWith(“UWL”); // g is the boolean value true

**SIMPLELIST**

1) SimpleLists are not random access (there is no such thing as an index into SimpleLists).
2) Assume that variable ‘x’ is of type SimpleList<E>. The following expressions are then meaningful: x.reset(), x.hasNext(), x.next(), x.add(e), x.remove(), x.size(). Note that the following expressions are not meaningful: x[i], x.length(), and x.remove(i).
3) Don’t forget to reset.
4) Most SimpleList processing will look similar to the following code:

   i) public boolean listContains(SimpleList<String> list, String item) {
       (1) list.reset();
       (2) while(list.hasNext()) {
           (3) String val = list.next();
           (4) if(val.equals(item)) return true;
           (5) }
           (6) return false;
   ii) }

**CONTROL FLOW**

1) An ‘if’ statement is used to make a choice between 1 or more alternatives.
2) A ‘for’ loop is used to repeat a chunk of code N times were it is possible to write some expression that represents the exact number of repetitions to execute. This is known as a ‘definite’ loop.
3) A ‘while’ loop is used to repeat a chunk of code some number of times. It is either impossible or in inconvenient to compute in advance how many times the loop will repeat. This is known as an ‘indefinite’ loop.
1) There are two very different ways of asking the questions “is X equal to Y”.

2) The ‘==’ notation asks if X is equal to Y under ‘identity equality’. In this case, X is only equal to Y if X and Y are precisely the same object. In other words, an object is equal only to itself and never another object.

3) The ‘equals(..)’ method asks if X is equal to Y under ‘content equality’. In this case X is equal to Y if they are objects of the same type and have the same internal state (or internal content).

4) Examples:
   a) String x = “Green Bay Packers”;
   b) String y = “Green Bay Packers”;
   c) boolean b1 = x == y; // b1 is false
   d) boolean b2 = x.equals(y); // b2 is true
   e) boolean b3 = x == x; // b3 is true