Explanation of Java

- **What is an object?**
  - An object is a value of a class type
  - An example is: String, Sound or a Robot

- **What is a primitive type?**
  - A primitive type is a number type or Boolean type (not an object)
  - They include char, int, double, Boolean
  - int and double are whole numbers or (possibly) decimals, respectively
  - Boolean only has two values (true or false)

- **What are the operators in java?**
  
  **For int or double:**
  
  + additive operator (also used for String concatenation)
  - subtraction operator
  * multiplication operator
  / division operator
  % modulus (remainder) operator // i.e. 9%5 = 4

  **For boolean:**
  
  == equal to
  != not equal to
  > greater than
  >= greater than or equal to
  < less than
  <= less than or equal to

- **Statements:**
  
  - Declaration introduces a variable by specifying its type and name. The declaration statement consists of two things:
    
    variableType - such as String, int or double
    
    variableName – an arbitrary name chosen by the programmer, it can be a single letter or a word but it does not have spaces;

    For example:
    
    String help;
    int a;
    double b;

  *The type of a variable designates what is allowed to be done to it and what it is allowed to do.

  But before you can use a declared variable, it must be initialized to a value.

  - Initialization gives a variable a value, using the assignment operator:
    
    - Remember that declaration always precedes initialization. Examples of an initialization statement (continuing with the variables from the above section):
      
      help = “ help”;

Created by Deborah Nelson 4/8/2007
- a = 9%2;
- b = 4.5 + a;
- String c = “Good job!” + help;

*Notice that the initialization statement consists of three things:
  *VariableName* - the name given the variable during declaration
  = - Assignment operator
  *value* - depending on the type of the variable; for example, String would
  be set to a value within quotations “ ”. int and doubles can be numbers

*Notice that a declaration and initialization statement can be combined, as in the
last example.

Important things to know in a declaration/initialization statement:
*These statements all end in a semicolon
*A variable name starts with a lowercase letter
*Only declare a variable once. When you refer to the variable again (after
declaration), do not put the *variableType* in front of the *variableName* – that is know as
re-declaring a variable and it should not be done.
* You can change the value of a variable as many times as necessary.
* For example, incrementing and decrementing statements change the value
  b = b + 1;
  a = a -1; //remember the short cut is to not use the assignment operator and just write
  b++; or a--;

• **Arrays**: An array is a collection of elements (objects or primitive types) that are
  all of the same type.

To create an array, use the following syntax:
```
elementType[] arrayName = new elementType[arrayLength];
```

For example, to create an array of integers named list:
```
int[] list = new int[5];
```

*The *arrayLength* of an array is fixed at the time of creation (and it is an int value); the
brackets are required; *arrayName* is arbitrary. The type of possible elements include:
String, int or double

To find the number of elements in an array and store it to an int variable, use the syntax:
```
int value = arrayName.length;
```
For example, *len* would be set to 5 with this statement:
```
int len = list.length;
```

Each element in the array has a specific location known as the *index*. (you can use a
variable to refer to the index). The first element in the array is at index 0. The last
element is at index *arrayName*.length – 1

To access an element, and store the value to a variable use the following syntax:
```
int value = arrayName[index];
```
//index should be a number or an int variable
For example, if list (created above) were an array containing {1,2,3,4,5} then
int n = list[3]; would set n to 4

- **Strings**: a String can consist of letters, numbers, spaces, etc
To create a String object:

```java
String stringName = "whatever";
//remember stringName is arbitrary
```
For example:

```java
String c = "bee";
```

To find the length of the string:

```java
stringName.length();
```
For example:

```java
int len = c.length(); //len would be 3
```

*Each letter in a string has an index, including spaces. Just as with arrays, the first
character of a string is at index 0 and the last character in the string is at the index
(stringName.length() – 1)

*Methods you should know that can be called on String objects:

1) **equals(String s)** // s is a parameter
   This method returns true or false.
   For example with the following strings:
   ```java
   String a = "A";
   String b = "Aa";
   
   Calling the method:
   a.equals(b) will return false
   ```

2) **indexOf(String s)** // s is a parameter
   This method returns the index of the first occurrence of the specified substring.
   Remember the index of the first character is 0. For example, with the following:
   ```java
   String e = "e";
   String d = "help";
   
   Calling the method:
   d.indexOf(e) will return 1
   ```

3) **substring(int n)**; // n is a parameter
   This method returns a substring that begins at the specified index.
   For example, with:
   ```java
   String whole = "help me";
   
   Calling the method:
   String sub = whole.substring(5); // String sub contains “me"
   ```

4) **substring(int m, int n)**; //m is the beginning index and n is the ending index
   This method returns a substring that begins at the specified index and ends
   at the specified index - 1.
   For example, with the following:
   ```java
   String w = "help me";
   
   Calling the method:
String s = w.substring(1,4)  // s contains “elp”

*Notice that when you call the method, you do not put the type ( int or String) in front of the parameter names

•  Classes and Methods

All of the Java above is used within methods, which are found within classes

When writing Classes and Methods:

•  The Class header in a class declaration consists of three things:
  1) public (accessor Specifier)
  2) class – required word
  3) className – an arbitrary name beginning with an Uppercase letter

    For example:

    public class OneHeapNim {
      //body of class
    }

•  The Method header in a method declaration consists of four things:
  1) Public (accessor Specifier)
  2) returnType (either: void, int, double, Boolean, elementType[ ], String)
  3) methodName – an arbitrary name beginning with a lowercase letter
  4) parameters (which are found within the parentheses)

    For example:

    public int makeMove(int numObjects){
      //body of method
    }

*notice that class and method headers do not end in semicolons; instead the body of the class and the body of the header is enclosed in brackets

* To invoke/call a method, use the format:

    objectName.methodName(parameters);

    for example, for Robot herbie:

    herbie.turnLeft();

If the method has a return type, set the returned value to a variable

    returnType variableName = objectName.methodName(parameters);

    for example, for String a:

    int len = a.length();

  o  Parameters

A method name is always followed by parentheses. The purpose of parentheses is to include the parameters. If a method does not have parameters, it is followed by empty parentheses – as you saw with the Robot method turnLeft();

Parameters refer to the list of variables in a method declaration. Each parameter has a variableType and a variableName. For example, numObjects is the int parameter of the following method
public int makeMove(int numObjects) {
}

*You do not need to declare your own parameters, then will be given to you. Also, don’t redeclare them. Use them as the method directs you to (For example, in the changeVolume(double factor) method in your Sound class, factor was the parameter by which you change the value of each SoundSample)*

Arguments are the actual values that are passed in when the method is called. When you invoke a method, the arguments used must match the declaration’s parameters in type and order. makeMove(3); is an example of calling the method

- Return statements
According to the method header, every method has a return type. If the method does not return anything, it has the return type void. For example,
The method header for the method turnLeft() is:
    public void turnLeft() {
    }
A method whose return type is not void must have a return statement that returns a variable/value consistent with the returnType specified in the method header.

Return statements consists of two things
1) return - this is required
2) expression – depends on the return type;

Example of return statements:
For the squareLength APT (with parameter str) the return statement could have been:
    return str.length() * str.length();
For the oneHeapNim APT the return statement could have been:
    return 1;

- If statements
If statements consist of an expression (using the Boolean operators listed in the third bullet) and a body enclosed within brackets. The expression statement does not end in a semicolon. If the expression is true, then the instructions within the brackets will be read. The expression is only read once.
The possible syntax of if statements include:
- if (expression) { statement(s); }
  OR
- if (expression){statement(s); }
  else {statement(s) }
  OR
- if (expression) { statement(s) }
  else if(expression) {statement(s) }
  else { statement(s) }

For example:
1) The variable numObjects is a parameter:
   if(numObjects%3 == 1) {

return 1;
}
else {return 2};

2) The array list holds ints; index is an int variable; max is an int variable
   if(list[index] >= max){
     max = list[index];
   }

- While loops

The while statement continually executes a block of statements while a particular condition is true. Its syntax is:
   while (expression) {
     statement(s);
   }

The while statement evaluates expression, which must return a boolean value. If the expression evaluates to true, the while statement executes the statement(s) in the while block. The while statement continues testing the expression and executing its block until the expression evaluates to false.

The steps of a while statement are to:
1) before the loop: Declare and initialize a variable that will be evaluated in the expression
2) within the parenthesis is the: expression
3) within the body: manipulate or access your elements (based on your problem)
4) within the body: increment or decrement the variable evaluated in the expression

For example to sum up the integers in an array named list:

```java
int j = 0;
int sum = 0;
while(j<list.length)
{
   sum = sum + list[j];
   j = j + 1;
}
```

- For loops perform the same function as a while loop but the three steps are all in one line (replace statement(s) with whatever you are supposed to do to each element)

```java
for(int j = 0; j<list.length; j = j + 1) {
  statement(s);
}
```

- For-each loops are useful if you are performing the same operation on each element. In the case below, elem will be set to the next value in list every time through the loop

```java
for(int elem : list) {
  statements(s);
}
```

*These statements do not end in semicolons! Instead two brackets { } designation the body of the statement. (However, statements within the bracket do end in semicolons)

* In debugging with loops, make sure your loop is not infinite (at some point, the expression must return false) and make sure that you do not go out of the bounds of your array or String - or whatever you’re looping through

- How to solve an APT
1) Read the instructions that explain the purpose of the APT
2) Solve the problem (without using code). Use test data if it helps
3) Before you write code, understand the parts of the method header
   a. Identity the return type, and what the method gives you (the parameters)
4) You can not make changes to the method header or the class header (unless
directed to do so by your TA)
5) Translate the solution into code. Ask yourself questions like:
   a. Does this solution require conditions (if statements)?
   b. Does it involve finding the sum, counting objects, etc in an array or
      String? (loops)
6) Believe you can solve the problem
7) If you don’t solve it, submit something that compiles & remember a return
   statement!