Java Language Basics

Program Structure
The basic structure of Java programs:
public class ClassName
{
    public static void main(String[] args)
    {
        program statements
    }
    user defined methods
}

Explanations:
• Public classes are accessible from any class.
• Classes, which are not declared public are accessible only within their package
• Public methods may be called from any class.
• Static methods do not operate on objects.
• The keyword void indicates that the method does not return a value.
• The array args of the main method stores command line arguments.

Source Code, Compilation and Execution:
The name of the source code needs to be the same as the name of the public class. There can be only one public class in a Java program. The source code needs to have the extension .java. During compilation the Java compiler produces the bytecode file. The bytecode file is given the extension .class. The Java interpreter starts execution when we type command java followed by the class name.

Data Types
Integers: int, short, long, byte
Floating-Point Types: float, double
The Character Type: char
The Boolean Type: boolean (values: true, false)

Operators
Arithmetic Operators: +, -, *, /, %
Relational Operators: ==, !=, <, >, <=, >=
Logical Operators: &&, ||, !

Strings
Strings are a standard class in Java. Constant strings are enclosed in double quotes.
String Concatenation: +
Selected Member Functions:

- int compareTo(String other)
  returns a negative value if the implicit argument comes before the explicit argument (in alphabetical ordering), a positive value if the explicit argument comes before the implicit argument, and 0 if the strings are equal
- boolean equals(Object other)
  returns true if the implicit argument equals the explicit argument
- int length()
  returns the length of the string

Variables

Variables in Java need to be declared. You need to initialize variables before you use them.

Examples:

```java
int n=5;
System.out.println(n);

String s=”Hello”;
String t=”world”;
System.out.println(s+” “+t);
```

If/else Statement

Both if and if-else statements are similar as in C++. As in C++ curly braces are not necessary if they enclose a single instruction.

```java
if(…)
{
    ...
}

if(…)
{
    ...
}
else
{
    ...
}
**While Loops**
While loops are similar to C++ as well. Again curly braces are not necessary.
```java
while (...) {
    ...
}
```

**For Loops**
For loops also follow the C++ pattern.
```java
for(initialization; condition; update) {
    ...
}
```

**Arrays**
Arrays are a standard class in Java. You declare an array by specifying the array type followed by the symbol `[ ]`. Objects of the Array class are created by an application of the operator `new`. You may explicitly list all elements of an array in a list enclosed by curly braces and separated by commas. If you copy one array variable into another, then both variables refer to the same array. As in C++ and Perl array indices run from 0 up to the length of the array minus one. Use `arrayName.length` to find the length of the array.

Selected Member Functions:
- static void sort(type [] a)
  Argument a is an array of a type listed in Data Types.
  sorts an array using a QuickSort algorithm
- static int binarySearch(type [] a, type v)
  Argument a is a sorted array of a type listed in Data Types, argument v has the same type as the elements of a.
  applies the BinarySearch algorithm to search for v
- static boolean equals(type [] a, Object other)
  Argument a is an array of a type listed in Data Types, argument other is an object.
  returns true if other is an array of the same type as a, if it has the same length, and if the corresponding elements match

Example:
```java
int n=5;
int[] numbers = new int[n];
for(int i=0; i<numbers.length; i++)
    numbers[i]=n-i;
Arrays.sort(numbers);
for(int i=0; i<numbers.length; i++)
    System.out.println(numbers[i]);
```
import java.util.*;

class SortJava {
    public static void main(String[] args) {
        int n = 5;
        int[] numbers = new int[n];

        for (int i = 0; i < numbers.length; i++) {
            numbers[i] = (int) (Math.random() * n);
        }

        System.out.println("Random numbers before sorting");
        for (int i = 0; i < numbers.length; i++)
            System.out.println(numbers[i]);
        System.out.println();

        System.out.println("Random numbers after sorting");

        Arrays.sort(numbers);
        for (int i = 0; i < numbers.length; i++)
            System.out.println(numbers[i]);
    }
}

Java Sort Method of Arrays Class
The code is contained in the directory java/sort on dunx1.
Quick Sort Algorithm in Java

The code is contained in the directory java/quick_sort on dunx1

Program’s structure

public class QuickSortJava
{
    public static void main(String[] args)
    {
        ...
    }

    public static void sort(int[] v, int left, int right)
    {
        ...
    }

    public static void swap(int[] v, int i, int j)
    {
        ...
    }

    public static int rand(int left, int right)
    {
        ...
    }
}

public class QuickSortJava
{
    public static void main(String[] args)
    {
        int n=5;
        int[] numbers = new int[n];

        for(int i=0; i<numbers.length; i++)
            numbers[i]=(int)(Math.random()*n);

        System.out.println("Random numbers before sorting");

        for(int i=0; i<numbers.length; i++)
            System.out.println(numbers[i]);

        System.out.println();

        sort(numbers,0,n-1);
    }
}
System.out.println("Random numbers after sorting");

for(int i=0; i<numbers.length; i++)
    System.out.println(numbers[i]);

public static void sort(int[] v, int left, int right)
{
    int i, last;
    if(left>=right)
        return;
    //move pivot to the v[left]
    swap(v, left, rand(left, right));
    last=left;
    //partition
    for(i=left+1; i<= right; i++)
        if(v[i]<v[left])
            swap(v, ++last, i);
    //restore pivot element
    swap(v, left, last);
    //recursively sort each part
    sort(v, left, last-1);
    sort(v, last+1, right);
}

public static void swap(int[] v, int i, int j)
{
    int temp;
    temp=v[i];
    v[i]=v[j];
    v[j]=temp;
}

public static int rand(int left, int right)
{
    return left+(int)(Math.random()*(right-left+1));
}