تمہیں معلوم نہیں کہ آسمانوں اور زمین کی بادشاہت خدا ہی کی ہے، اور خدا کے سوا تمہارا کوئی دوست اور مددگار نہیں۔ البقرۃ آیت ۱۰۷

A person who never made a mistake never tried anything new. Albert Einstein

Lecture 06

Type of Codes, Errors, More Library Functions and Programs

In this lesson we will discuss type of codes and errors followed by introduction of some more functions and finally programs to do practice.

Type of Codes

Here we are categorizing Java code with respect to writing and execution of programs. There are two types of codes:

- **Source Code:** A program written by humans (normally) and visible in the form that is human readable is typically called a source code. Some editor is used to write source code like Notepad, Wordpad, Textpad, Eclipse, NetBeans etc. Interestingly source code has a form which can't be run on any computer. A source code can be read, modified, extended or deleted but to execute a Java program, source code must be compiled. Java compiler is included in **JDK** (Java Development Kit is a bundle of softwares required for Java development.
- **Byte Code:** Java byte code is generated by Java compiler. Java compiler itself is a program that takes source code as **input** and generates byte code as **output** if there is no syntax errors exist in code. Java compilers are plat-form dependent whereas Java programs are plat-form independent seems interesting. Java has a slogan "Write once run everywhere". Once a program is compiled source code is not required to run the program. Rather you may take byte code anywhere and execute. **JRE** (Java Runtime Environment) is required to run Java byte/ Java program.

  Note: JDK normally includes **JRE**.

Type of Errors

Again we are discussing one typical category of errors, errors may be categorized with some other consideration:

- **Syntax / Compile Time Error:** Every programming language has its own syntax (Grammar of language) that is rules to write programs. Though to experienced programmers programming languages seems not very different but it is fact that there is variety of syntax exist. There are many similarities in syntax of C++ (a language used in academia since long) and Java, yet there are some differences. Like you may write int x=3.5 in C++ but not allowed in Java, similarly C++ allows if (a=5) whereas Java does not allow. Java compiler check syntax before generating bytecode, if there is any syntax error bytecode is not generated and compiler tries to identify the error at its best (though many times it fails to identify exact error but once again with experience developers can identify syntax error also sophisticated editors like Eclipse, NetBeans identify syntax errors while writing codes. Some common syntax errors are:
  - Java is **case sensitive** therefore using small letter instead of capital letter or vice versa. This errors is for both Java key words and variables
  - Java is **strict type** therefore assigning incompatible type to variables like int t=3.56
  - missing semicolon at the end of the statement
  - writing = instead of ==, typically inside if condition or loop conditions
  - missing operator or operand in expression

- missing any type of brackets (Large, Middle, Small)
- using variable without declaration
- writing variable on right hand side of expression without initialization
- using a class/ function not in **lang** package/ folder without **import**

Exception / Run Time Error: Some people consider Run Time Error as Logical Error, which is to me different. If during the execution of program some error occurred and exception is generated which if not handled can crash the program, means your program may terminate immediately. Exception handling is a complete topic in high level languages and appropriate handling is not a scope of this course, though we may do it at some stage. Some examples of run time errors are:

- a typical example is divide by zero, typically done using variables and during run time variable may have 0 value
- function is called to take integer input but real values are entered or characters (other than digits) are entered
- array (a variable having list of values) is accessed beyond the limit. For example if array size is 5 and code is written to access 6th value. This is by programmers mistake to give wrong value to index of array
- a file is to be opened for reading whereas file does not exist or have different name or rights are not given to read file

Logical Error: Logical errors cause unexpected or wrong output. Though these are normally done by mistake of developers but depending upon the nature of programs sometimes it is really hard to find them. A process called debugging is used to find logical error. There are numerous possibilities for logical errors, some examples are:

- assigning wrong value to variables. Very common is missing digit or character, disposition writing 56 instead of 65.
- value exceeds variable range
- variable is not re initialized. Like if sum is used to calculate one set of values and same sum variable is used for another set of values, it must be initialized to 0 before second summation
- similarly loop variables are not initialized in second loop
- or there may be altogether wrong logic is used. For example to sort elements only few numbers are sorted.

Summarizing among these syntax errors are easiest because same syntax is used again and again and also compiler is your friend to find them. At second number are Run Time Errors. They are easier to detect but problem with them is there may arise a situation after long when a typical run time occur. For example network connection breaks down. The most difficult are logical errors, unfortunately no tool available to find them but luckily there are development softwares like Eclipse and NetBeans which helps a lot in **debugging**. Step by step execution, watching variables step by step are some of the salient features of debuggers.

Functions and Programming Practice

As we already introduced functions in previous lectures, here we will discuss how we can use related functions to do a task. For example a String may have many words like "This is OOP class". Now if we want to separate these words, we may use **substring** function but substring function requires start and end arguments; whereas; for first word I know start is 0 but what is end, if I say it is 4, it means code will work for only strings having first word of length 4. Therefore, to generalize the code I need to find position of space using **indexOf** function which gives position first occurrence of any character inside string. See following code:

```
String s = "Where are you";
System.out.println("Position of space is:" + s.indexOf (' ') );
```

The answer will be 5 in this case; whereas if we write same line with previous string the answer will be 4. Therefore, we may write a code to split all words in previous string using **indexOf** function and substring where **indexOf** name has two functions one has single argument; whereas; second has two arguments, one is character to find and second is starting position. Now see the code:

```
String s = "This is OOP class";
int start = 0, end = s.indexOf (' ');
System.out.println( s.substring(start,end) );
start = end, end = s.indexOf (' ', start + 1);
System.out.println( s.substring(start,end) );
start = end, end = s.indexOf (' ', start + 1);
System.out.println( s.substring(start,end) );
start = end, end = s.indexOf (' ', start + 1);
System.out.println( s.substring(start,end) );
```

You may have question program is again hard coded, yes I agree though it can work for all strings having 4 words, however, we can handle any string using loops. Within few weeks we will be there. Function can also be used as arguments of other functions. For example in above code we have written substring function inside **print** function. Like to compare two strings without case that is spelling must be same ignoring upper or lower case, we may convert both strings in upper or lower case like:

```
String s1 = "kamal ahmad";
String s2 = "kamal Ahmad";
s1.equals(s2);// false because letter case mismatch
s1.toUpperCase().equals(s2.toUpperCase());//true
```

Programming Practice

Next we will do some practice by using random functions from Math class. This function has no argument and value is unpredictable like following code has output:

```
class RandomFuntions{
	public static void main(String []args){
		System.out.println(Math.random());
		System.out.println(Math.random());
		System.out.println(Math.random());
	}
}
```

```
0.6674039080118588
0.3163673793562789
0.5818845067234999
```

If you run again you may get some other output between 0 and 1. Somehow this random number function is very useful to do test run with different values. It saves you from giving input from keyboard that is ofcourse tidy job. However an obvious problem is the value returned by random function. Here we will address this problem that how with little arithmetic we can obtain value within required range.

First of all multiplying a random number with n will change the range from **0 to 1** to **0 to n**. Moreover addition with k can change range to k to n+k. It doesn't matter whether k is positive or negative. Lastly using type casting we can convert values to integer whereas originally it is double. Now see

```
System.out.println(Math.random()*50);//32.38250752365021
System.out.println(Math.random()*50);//24.035292753202675
System.out.println(Math.random()*50);//35.4232354648898
```

Here we have multiplied with 50 therefore you can expect any value between 0 and 50. Therefore, you can get any number between 0 and n by multiplying with n. However, in following code you can see multiplication factor is 100 and additive factor is 50, hence output is between 50 and 150. Like there is one value 53 close to 50 and another value is 141 close to 150. Though it not necessary to find values always like this. There may be all values less than 100 or above 100 or anything else.

```
System.out.println(Math.random()*100+50);//53.870502035235255
System.out.println(Math.random()*100+50);////84.32247390443526
System.out.println(Math.random()*100+50);//141.0377212554997
```

Lastly these values are double whereas we may need integer values in many cases, therefore, type casting is required for example see the code:

```
System.out.println((int)(Math.random()*100+50));//128
System.out.println((int)(Math.random()*100+50));//134
System.out.println((int)(Math.random()*100+50));//114
```

A last thing very important is type casting should be done carefully otherwise you may get incorrect values like:

```
System.out.println((int) Math.random()*100);//0
```

In above code result will be 0 because expression is solved left to right, hence (int) will operate on Math.random() and give 0 because value is between 0 and 1 hence fractional part will be chopped and remaining is 0, however to handle this we will use parenthesis and multiply before casting, by this casting operation will be performed after value goes above 1 and hence we may get integer portion in result. Lastly see following program where we have replaced key board input by random function.

```
class DistanceConvertor{
      public static void main(String []args){
            int km=(int)(Math.random()*100+500);
            double miles=km/8.0*5;
            System.out.println(km+" kms equivalent to "+miles+" miles");
      }
}
```

Sample runs of above code are:

```
128 kms equivalent to 80.0 miles
105 kms equivalent to 65.625 miles
119 kms equivalent to 74.375 miles
```