CS102
Introduction to data structures, algorithms, and object-oriented programming

DAY 3
## Summary of Java Primitive Types

<table>
<thead>
<tr>
<th>TYPE NAME</th>
<th>OPERATORS</th>
</tr>
</thead>
</table>
| int       | * / + - %
            | += -= *= /= %=
            | ++ -- |
| double    | Same as above except for % ops |
| boolean   | ! (not)
            | && (and)
            | || (or)
            | <, >, <=, >=, !=, == |
| char      | A char is stored as a number, so any arithmetic or relational operator can be used on them. |
A class may contain so-called “class members”, which can be variables (fields) or methods (actions).

A class (aka global) variable (aka field) is declared outside any class methods, but inside the class boundaries {}. It is accessible in any method of the class. Local variables are declared inside methods.

A class method is a static method declared inside the class boundaries.

Before you start to write new classes (data types), you will use many built-in types.
The “dot” operator

When you see the “.” in an expression, e.g.:
   System.out.println();

It means you are accessing a field or method inside a class.

The statement System.out.println(); means you are calling the println method, with no arguments. The println method is a member of the out field of the System class.

If you look up the System class in the Java API, you’ll find that the out field is of type PrintStream and if you look in the PrintStream class, under the “method summary”, you’ll see many versions of the println method. This is known as overloading.
Overloading method names

A method in a class is overloaded when there are other methods in the class with the same name and same return type, but with a different number or type of arguments.

There are 10 different versions of println in the PrintStream class, including one with 0 arguments.

At least one primitive operator is also overloaded. The + operator can be used to add numbers or to concatenate Strings with other data types.
String (Eck 2.3.3)

String is a class, and a value of type String is an object.

A String object contains data, namely the sequence of characters that make up the string. It also contains methods.

String functions, unlike System and Math functions, are called on objects of type String, not on the class:

```
String myCat = "Ooma"; // declare and initialize myCat
// myCat.length() = 4
// "I love Java".length() = 11
```
Patterns of Method calls

Method calls always have a set of ()’s after the name.

If the method you are calling is *static* you start the call with the class name:

```java
System.out.println(); // no arguments
Math.max(5, 12);       // 2 arguments
```

If the method you are calling is in an object, you start the call with the object name. Objects are always variables, so they never begin with a capital letter:

```java
String favoriteColor;
favoriteColor = “green”;
System.out.println("Your favorite color is “ +
                    favoriteColor.toUpperCase());
```
Different ways to read input

Java arrays:
One simple way to get input is from the command line. When you type anything after typing “java FileName”, the items are input as whitespace-separated entries in an array of String type.

The (String[] args) part of the main method signature is filled by whitespace-delimited Strings that you enter after the FileName. See AddThree.java.

Notice that AddThree uses the static method parseInt from the Integer class to convert the strings to ints.
Java Input

As Eck puts it, in Section 2.4.2 of his book:

“For some unfathomable reason, Java has never made it very easy to read data typed in by the user of a program.”

Because of the complexity of getting data input in a program, most book authors write their own libraries of input functions to make it easier for us to input data.

The input library provided by Eck is `TextIO.java`

To use the static functions in `TextIO`, you need to copy the `TextIO.java` file into the same directory as the file you are writing. A link to `TextIO` API is on our website.
packages and import statement

Packages in java consist of a compressed file of classes available as bytecode. As you will learn in lab today, the only package from the Java Developer’s Kit (JDK) that is automatically opened is the java.lang package.

To use any of the classes included in another package, you need to put an import statement at the top of your program before the class declaration line.

Today in the second part of lab, you will import the package javax.swing to allow you to access a graphical method of input for your java program.

```
import javax.swing.*;
```

Import statements are also followed by ;   The * means, access all files contained in the swing package.
packages and fqn’s

The class you will use in lab today is from the javax.swing package and it is called JOptionPane.showInputDialog(...). Import statements save the programmer from typing and re-typing the package name. But all of the packages are available to us.

String a = JOptionPane.showInputDialog("Enter value a : ");

Another way to use classes written in other java packages is by invoking the method with its “fully qualified name” (fqn). In this case, you would use the following line to ask for input of a String from the keyboard:

String a = javax.swing.JOptionPane.showInputDialog("Enter value a : ");
Lab 2

In lab today, you will be downloading the TextIO.java class and a starter file. You will follow the lab instructions posted on our website to modify the starter file.

Statements:
 a. variable declarations,
 b. assignments,
 c. conditionals (if, else), **
 d. loops (for, while, do-while), **
 e. method calls,
 f. returns, and
 g. imports.