Revision: Variables, Expressions and Mathematical methods

- Variables provides a way in Java to *use the memory*.
- Constants are used to *express values directly* in Java.
- Operators and methods join variables and constants together to form expression.
- An expression expresses a computation. Simple arithematics are done by operators, and more advanced computations are done by mathematical methods.
- An expression has a type and a value. The type of the expression is *determined completely* by the type of variables and constants used.

Using inputs

- Unlike most other languages, Java input system is very complicated.
- Most programmers will **use a custom library** for performing input.
- The author of our textbook written a library, which can be used for education purpose. We will use this library for our course.
- This means we will use the Java programs **written by others**.
- This involves installing a package, and then use a few more statements in your programs.
- Installation will be covered in the workshop.

Example program

```
/* Purpose: solve a quadratic equation. */
import chapman.io.*
public class Quadratics {
  public static void main(String[] args) {
    StdIn in = new StdIn();
    System.out.print("a? ");
    double a = in.readDouble();
    System.out.print("b? ");
    double b = in.readDouble();
    System.out.print("c? ");
    double c = in.readDouble();
    double determinant = b*b-4*a*c;
    System.out.print("The first solution is ");
    System.out.println((-b+Math.sqrt(determinant))/2/a);
    System.out.print("The second solution is ");
    System.out.println((-b-Math.sqrt(determinant))/2/a);
  }
}
```


Using the library

- You must *tell the compiler* that you are using a custom library:

  import chapman.io.*;

- Then, you must call *new StdIn()* to get an object, which can then be used for input. You can save it in a variable, as follows:

  StdIn in = new StdIn();

- Then you can use the variable to read in a value.

  in.readDouble()
  in.readInt()
  in.readBoolean()
  in.readString()

- Like a Mathematical method, this gives you an expression, so you can use the value by assigning it to a variable, use it to build further expressions, etc.

Another example: the day of week

```
/* Purpose: Find the day of week. */
public class DayOfWeek {
  public static void main(String[] args) {
    int yyyy = 1997;
    int mm = 7;
    int dd = 1;

    int mm1 = 9 + mm;
    int yyyy1 = yyyy - 1;
    yyyy1 = yyyy1 + mm1 / 12;
    int m = mm1 % 12;
    int cc = yyyy1 / 100;
    int yy = yyyy1 % 100;
    int sum = 146097 * cc / 4
            + 1461 * yy / 4
            + (153 * m + 2) / 5
            + dd + 2;

    System.out.print("Day of week: ");
    System.out.println(sum % 7);
  }
}
```


Expressions with side effects

Note that the read*Type*() methods have a side effect: it consumes the input. Therefore, if you call readDouble() twice, you can get different results.

For this reason, the evaluation order of an expression is really important. Consider the following statements:

```
double d = in.readDouble() + 2 * in.readDouble();
```

What will happen if the user type the following?

```
2.5
-5.3
```