Increment and Decrement

- In programming, we often add one or subtract one from a variable of an integer type
- This is written ++ or - -

```
int x = 3;
x++;     // x is now 4
x--;     // x is 3 again
x--;     // x is 2
```

- Increment and decrement combine a computation and an assignment

Increment and Decrement: be careful

- Increment and decrement can be used in the middle of an expression

```
int x = 2;
if (x++ > 2) ...
```

- With x++, the value of x changes *after* its value is used in the expression
  - so this condition evaluates to false
  - the equivalent ++x changes the value of x *before* its value is used in the expression
- We will look at this more later – ICS 111 students often make mistakes when using ++
  - you are welcome to use ++ when you are learning, but encouraged to be very cautious if you use ++ on quizzes or assignments

Java Math Library

- Computers are good at math, we should be able to use them for more than +-*/%
- The Java Math library provides the most common math functions:
  - `double square = Math.pow(PI, 2);`
  - `double root = Math.sqrt(square);`
- and many more: sin, cos, tan, exp (e^x), log, and so on
- See here for a full definition:

  https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Math.html

Converting Between Integral and Floating Point Types

- Assigning an integer to a float is easy:

```
int three = 3;
double doubleThree = three; // doubleThree is 3.0
```

- Assigning a float to an integer always truncates:

```
double fourEight = 4.8;
int four = (int)fourEight; //four is 4
```

- The type in parentheses is a **typecast**, or simply a **cast**
- We can also round:

```
long five = Math.round(fourEight);
```

Strongly Recommended

- Do the self-check exercises at the end of Section 2.2 in the textbook
- Actually write the code. You can print a variable with

```
System.out.println (variable);
```

- For example, using a variable from the last slide, we can write

```
System.out.print (“five is “);
System.out.println (five);
```

Summary

- Computers are good at math
- Variable declarations must include:
  - initialization
  - standard naming convention
    - camel case for variables
    - all uppercase for constants
- A variable is in scope from its declaration to the end of the enclosing block

ICS 111
Input and Output (I/O), Strings

- Java Input
- Java Output
- Java Strings

Input and Output
I/O Devices

- A computer typically has a number of I/O devices, such as:
  - keyboard
  - mouse
  - display
  - speakers
  - microphone
  - camera
  - various two-way radios

Input and Output

- Computers / programs can read from a device to get input, and/or write to a device to produce output
  - at the computer level, it is hardware I/O
  - at the program level, it is software I/O
- I/O may be from/to humans, and then is often text, graphics, sound or video
- I/O may be from/to other machines, and then is usually binary

I/O Devices: Storage

- Rotating disks and flash drives are not usually considered I/O devices
- However, the computer treats them the same as I/O devices:
  - reads data from the devices
  - writes (stores) data on the devices

System.out.print

- You are familiar (from the Hello World program) with `System.out.println`
  - prints its argument, then a newline
  - can actually print multiple arguments, joined by +

    System.out.println ("hello " + "world");
- System.out.print is the same, but does not print the newline

Introduction to Java Text Input

- Text (a sequence of characters) can be used to represent numbers
  - for example the text "1234" represents the number 1234
- The Java library `java.util.Scanner` can convert user input (a sequence of characters) to numbers
- You must first declare a variable of type Scanner:

```
java.util.Scanner in =
  new java.util.Scanner(System.in);
```

Java Import Statement

- The full name for System.out.println is actually java.lang.System.out.println
  - Java automatically imports everything in java.lang
  - everything else can be imported explicitly

```
import java.util.Scanner;
...
Scanner in = new Scanner (System.in);
```

- The import statements usually go at the beginning of the file

Using the Java Scanner

```
import java.util.Scanner;
Scanner in = new Scanner (System.in);
System.out.print ("Enter number: ");
long value = in.nextLong();
System.out.println("value: " + value);
```

- nextInt(), nextLong(), nextDouble(), etc.
- if the input cannot be converted, crashes the program
  - try it at home!! Enter "abc" to a number scanner
- full documentation is at https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Scanner.html

More about Printing

- As well as print and println, Java (like many other languages) has a "formatted print", or `printf`
- printf takes as its first argument a format string
- the format string is printed as-is, except where % characters are
- each % character describes the format for one of the arguments after the format string

printf examples

```
printf("value is %d\n", value);
```

prints an integer value (%**d** for decimal)

```
printf("price $%.2f\n",dollars);
```

prints a floating point value with two decimal digits (%**f** for float/double)

- in each case, \n represents a newline
- you can also specify a field width:

```
printf("value is %3d\n", value);
```

adds spaces before printing any number < 100

printf exercise

- copy, paste (into a file TestPrintf.java), and run this program. Then modify it until you are sure you understand what it does and how it works

-
  ```
  class TestPrintf {
    public static void main (String [] args) {
      int value = 99;
      double dollars = 12.3456789;
      System.out.printf ("value is %d\n", value);
      System.out.printf ("price %.2f\n", dollars);
    }
  };
  ```
- self-test question: when only two digits of the fraction of a floating point number are printed, is the number rounded or truncated?

Java Strings

- a string is a sequence of characters
- strings in Java are written between "

```
String nextPlayer = "Alice";
...
nextPlayer = "Bob";
```

- like any other variable,
  - string variables must be initialized, and
  - string variables can be assigned to
- note the Java type `String` always begins with an uppercase S

String Concatenation

- Java uses the same operator for String concatenation as for addition: +

```
String couple =
   "Harry " + "and " + "Megan";
```

- Java uses the type of the operators to decide whether to add or concatenate:
  - if at least one operator is a string, Java concatenates
  - if both operators are numbers, Java adds
- Java automatically converts numbers to strings when needed for concatenation
  - `System.out.print (100 + " students in this class");`

String Length

- The length of a string is the number of characters in the string

```
String name = "edo";

int nameLen = name.length();
```

- nameLen has the value 3

String Input

```
Scanner in = new Scanner (System.in);
```

`in.next()` returns the next word in the input

- for example, if the input is "hello world", the first call to in.next() returns "hello", and the second returns "world"

String Escape Sequences

- You saw that a backslash n is a newline
- that is, `\n` is an actual character
  - at runtime \n is a single character, even though in the source code it is written as two characters
- backslash can also escape quotes:

```
String greet = "say \"hi\"";
```

- greet.length() is 8 -- each \" is a single character

Characters

- A string is a sequence of characters
- English characters can be represented in a single byte (a number less than 256)
- for example, 'a' has the value 97, 'A' has the value 65
  - we use single quotes to enclose characters
- characters in other languages may need more than one byte
- the Chinese/Japanese character '江' has the value 27743
- the `charAt` method of `String` returns the character at a given position
  - character positions start at 0:

    ```
    String name = "edo";
    char d = name.charAt(1);
    ```

Substrings

- As well as string concatenation, sometimes you only want part of a string

```
String name = "edo biagioni";
String lastName = name.substring(4); // biagioni
String firstName = name.substring(0, 3); // edo
```

- character positions start with 0:
  - 'e' at 0, 'd' at 1, 'o' at 2, ' ' at 3, 'b' at 4...
- the one-argument substring method returns the substring from the given index to the end of the string
- in the two-argument substring method, the first argument is the starting position, and the second argument is the position after the end
- **Strongly recommended**: carefully study and understand the Initials.java example in Section 2.5.6 of the book

Dialog Boxes

```
import java.swing.JOptionPane;
String fromUser =
  JOptionPane.showInputDialog("enter number");
double value = Double.parseDouble(fromUser);
JOptionPane.showMessageDialog(null, "v: " + value);
```

- parsing means to look into a string to extract a value of a different type, in this case a double
  - if the string you enter does not represent a number, the program will crash at parseDouble
- full documentation is at https://docs.oracle.com/en/java/javase/11/docs/api/java.desktop/javax/swing/JOptionPane.html

Dialog Boxes Complete Example

```
import javax.swing.JOptionPane;

class TestDialogBoxes {
  public static void main (String [] args) {
    String fromUser = JOptionPane.showInputDialog("enter
number");
    double value = Double.parseDouble(fromUser);
    JOptionPane.showMessageDialog(null, "your number is: " +
value);
  }
};
```

- compile and run
- see what happens when you enter "hello world" instead of a number

Summary

- Strings are used to represent text
- length, charAt, substring methods
- strings are indexed starting with zero

- text input can be from console, or dialog boxes
- text output can be to console, or dialog boxes
- printf gives control over the layout of the text