First we welcome:

Duncan Hogg
Computer Science Co-op Co-ordinator

Office: ECS 230
Email: dshogg@uvic.ca

Java Tutorials

Tutorials coming (Thursday & Friday):

- 1-3 Today ECS 258 !!
- 1-3 Friday ECS 258

```
public class Circle {

  public static final double PI = 3.14159265359;

  public static void main (String[] args) {

      //program entry point

      double radius = 17.3;

      double circumference = 2*PI*radius;
      double area = PI * radius*radius;

      System.out.println("The circumference is "
                          + circumference);
      System.out.println("The area is " + area);

  }
}
```

Input from Keyboard

```
import java.util.*;
public class Circle {

  public static final double PI = 3.14159265359;

  public static void main (String[] args) {

      //program entry point
      Scanner in = new Scanner(System.in);
      double radius = in.nextDouble();

      double circumference = 2*PI*radius;
      double area = PI * radius*radius;

      System.out.println("The circumference is "
                          + circumference);
      System.out.println("The area is " + area);

  }
}
```

```
nextInt()
next()
```

Input from File

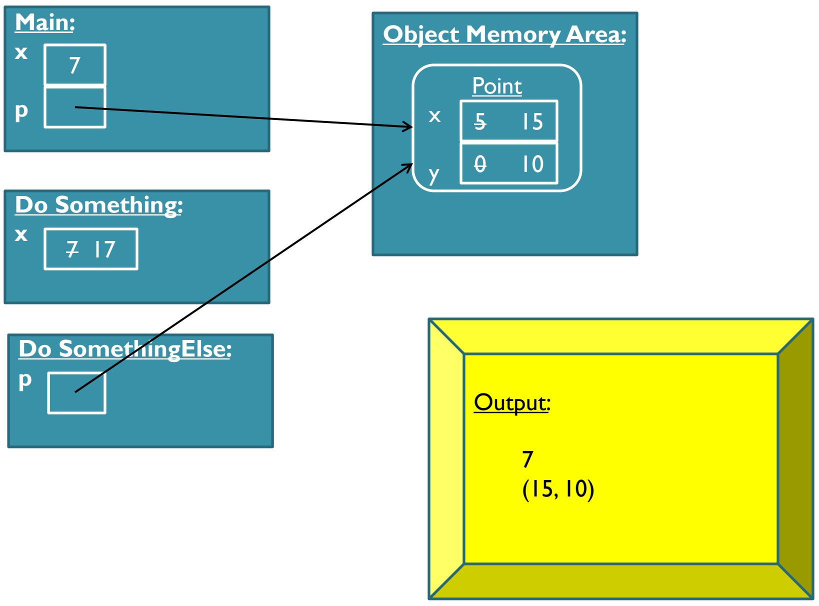
```
import java.io.*;
import java.util.*;
public class Circle {

  public static final double PI = 3.14159265359;

  public static void main (String[] args)
           throws FileNotFoundException  {

      //program entry point
      Scanner in = new Scanner(new File("name.txt"));
      double radius = in.nextDouble();

      double circumference = 2*PI*radius;
      double area = PI * radius*radius;

      System.out.println("The circumference is "
                           + circumference);
      System.out.println("The area is " + area);

  }
}
```

Quiz – From Class #1

```
class Point {
  public int x;
  public int y;
  Point (int theX, int theY) {
      x = theX;
      y = theY;
  }
  public String toString() {
    String s = "(";
    s+= x + "," + y + ")";
    return s;
  }
}
```

```
class quiz0 {
  public static void doSomething (int x) {
    x = x + 10;
  }
  public static void doSomethingElse (Point p) {
    p.x = p.x + 10;
    p.y = p.y + 10;
  }
  public static void main (String args[]) {
    int x = 7;
    Point p = new Point(5,0);
    doSomething(x);
    doSomethingElse(p);
    System.out.println(x);
    System.out.println(p);
  }
```

Memory Trace for Quiz 0's Execution



Output:

```
7
(15, 10)
```

Example 1: Write your Own Class

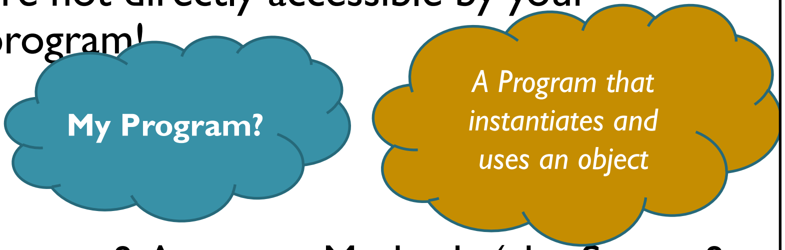
- Instantiate your own objects!

Example: the Patient

Code to be developed in Class

Mutator & Accessor Methods

When attributes are declared 'private' they are not directly accessible by your program!



Mutator & Accessor Methods (aka. Setter & Getter Methods) allow your program to change and view the object's attributes in a manner controlled by the programmer of the object's class.

Mutator & Accessor Methods

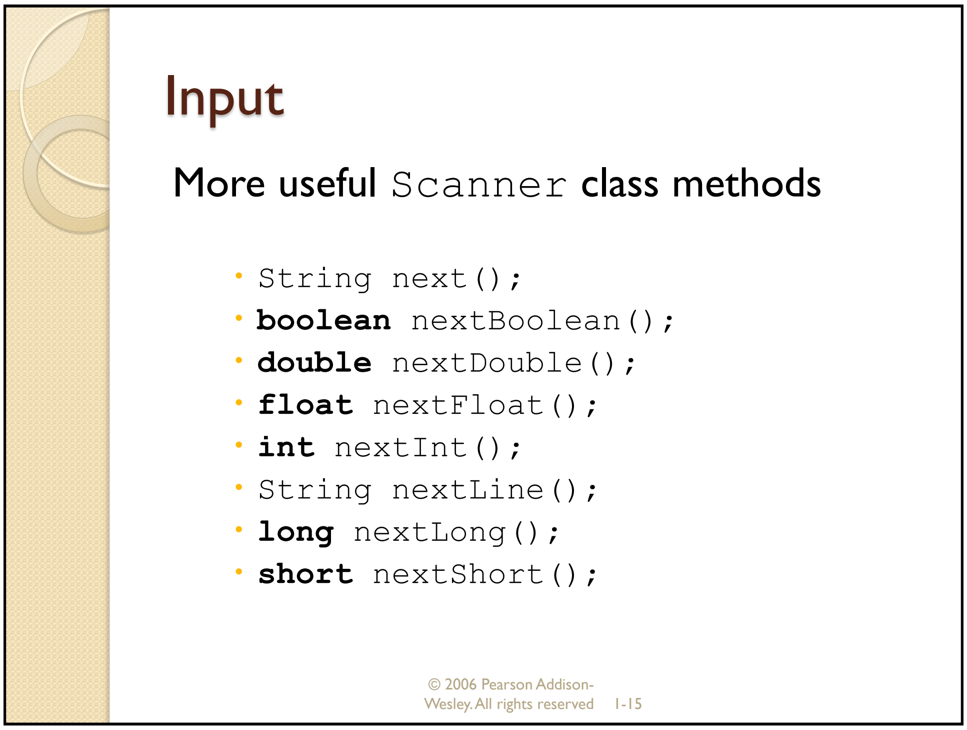
- Add Mutator & Accessor Methods to Patient.java
- Test those Methods using VictoriaHospitalSystem.java

Text Input and Output

- Input and output consist of streams
- Streams
  - Sequence of characters that either come from or go to an I/O device
  - `InputStream` – Input stream class
  - `PrintStream` – Output stream class
- java.lang.System provides three stream variables
  - `System.in` – standard input stream
  - `System.out` – standard output stream
  - `System.err` – standard error stream

Input: The `Scanner` class

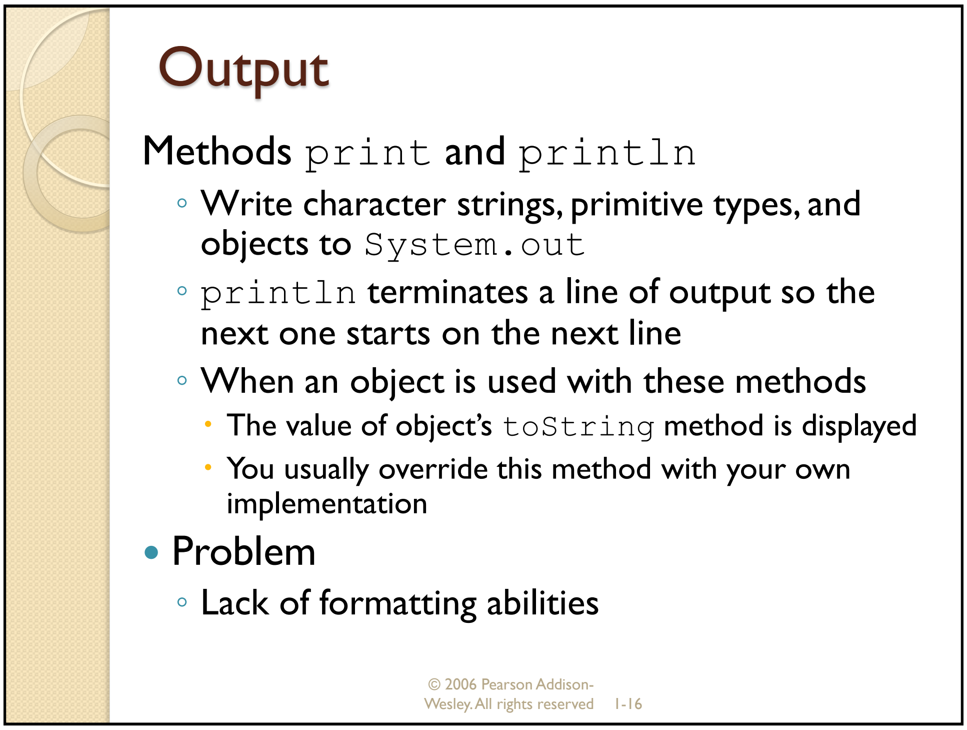
```
int nextValue;
int sum=0;
Scanner keyBoardInput = new Scanner(System.in);
nextValue = keyBoardInput.nextInt();
while (nextValue > 0) {
    sum += nextValue;
    nextValue = keyBoardInput.nextInt();
} // end while
keyBoardInput.close();
```



Input

More useful `Scanner` class methods

- `String next();`
- **`boolean`** `nextBoolean();`
- **`double`** `nextDouble();`
- **`float`** `nextFloat();`
- **`int`** `nextInt();`
- `String nextLine();`
- **`long`** `nextLong();`
- **`short`** `nextShort();`



Output

Methods `print` and `println`

- Write character strings, primitive types, and objects to `System.out`
- `println` terminates a line of output so the next one starts on the next line
- When an object is used with these methods
  - The value of object’s `toString` method is displayed
  - You usually override this method with your own implementation

- Problem
  - Lack of formatting abilities

Output

Method `printf`

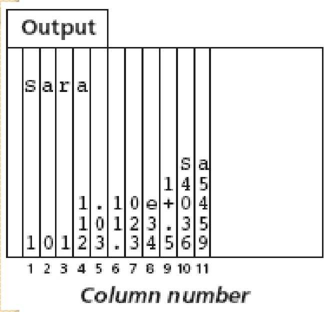
- C-style formatted output method

```
printf(String format, Object... args)
```

Example:

```
String name = "Jamie";
int x = 5, y = 6;
int sum = x + y;
System.out.printf("%s, %d + %d = %d",
                   name, x, y, sum);
//produces output Jamie, 5 + 6 = 11
```

Output



```
String name = "Sarah";
double y = 10123.34568;
int n = 145;
System.out.printf("%.4s\n", name);
System.out.printf("%10.2s\n", name);
System.out.printf("%10d\n", n);
System.out.printf("%10.2e\n", y);
System.out.printf("%10.2f\n", y);
System.out.printf("%5.5f\n", y);
```

Figure 1-10

Formatting example with `printf`

Text Files

- Designed for easy communication with people
  - Flexible and easy to use
  - Not efficient with respect to computer time and storage
- End-of-line symbol
  - Creates the illusion that a text file contains lines
- End-of-file symbol
  - Follows the last component in a file
- Scanner class can be used to process text files

Text Files

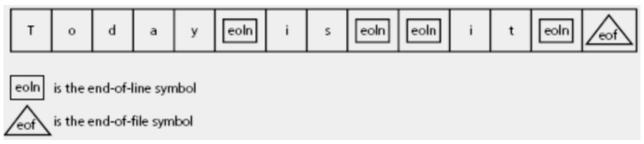


Figure 1-11

A text file with end-of-line and end-of-file symbols

Text Files

Example

```
String firstName, lastName;
int age;
Scanner fileInput;
File inFile = new File("Ages.dat");
try {
  fileInput = new Scanner(inFile);
  while (fileInput.hasNext()) {
      firstName = fileInput.next();
      lastName = fileInput.next();
      age = fileInput.nextInt();
      System.out.printf("%s %s is %d years old.\n",
                          firstName, lastName, age);
  } // end while
  fileInput.close();
} // end try
catch (FileNotFoundException e) {
  System.out.println(e);
} // end catch
```

Java Exceptions

- Exception
  - Handles an error during execution
- Throw an exception
  - To indicate an error during a method execution
- Catch an exception
  - To deal with the error condition