CSC115 Lecture 3

LillAnne Jackson
csc115@csc.uvic.ca
http://connex.csc.uvic.ca

Lectures: A01 MR 11:30 am-12:50 pm HHB 105

Office Hours: M 1:00-2:00 pm ECS 532
Th 10:00-11:00 am ECS 532

Did you the things that were to be done before this Class?

- Print and Read your course outline.
- **Get your CSC Account at** http://connex.csc.uvic.ca/
- Locate the Course website?
- Do as many exercises as possible from the end of Chapter 1.

Today:

- Assignment 1 Discussion
- More Classes & Objects
- Arrays

Assignment 1

- Connex CSC 115 Site:
  Resources, Assignments, Assignment 1
- Machine Marked:
  - Must use the *exact* file names specified!
  - Must produce the *exact* output formats!

Arrays

One-dimensional arrays (continued)

Initializer list example

```
double [] weekDayTemps = {82.0, 71.5, 61.8, 75.0, 88.3};
```

You can also an declare array of object references

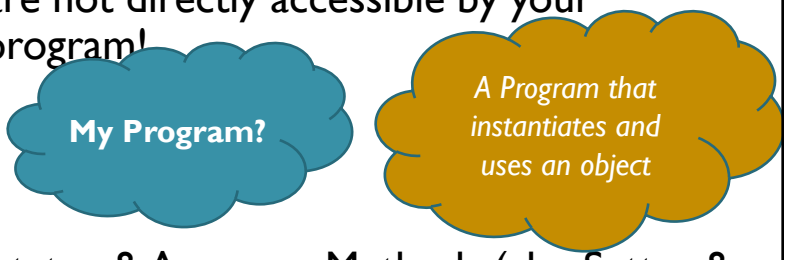
 1-5

Recall: Patient.java

- Developed on Thursday:
  see posted file: Patient.java
  - Contains data items (name, bloodType, etc) called *attributes*
  - Contains actions that can be performed on those attributes, called *instance methods*.
- Lets add a program that *instantiates* Patient.java objects:
  VictoriaHospitalSystem.java

Mutator & Accessor Methods

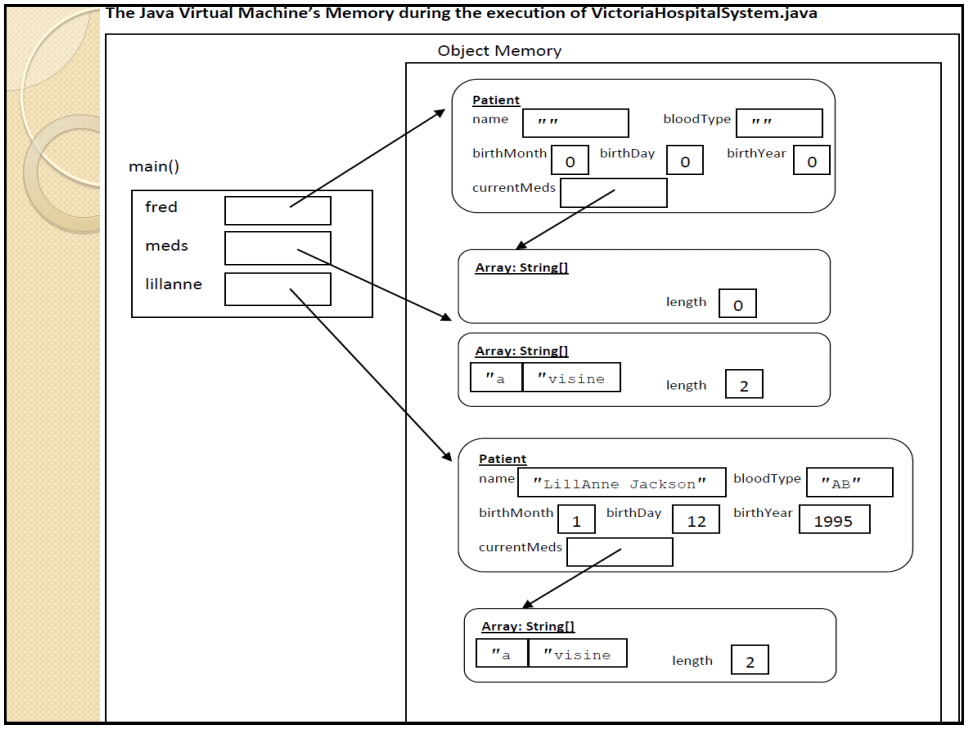
When attributes are declared 'private' they are not directly accessible by your program!



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



Useful Java Classes

- The `Object` class
  - Java supports a single class inheritance hierarchy
    - With class `Object` as the root
  - More useful methods
    - `public boolean equals(Object obj)`
    - `protected void finalize()`
    - `public String toString()`

Useful Java Classes

- String classes
  - Class `String`
    - Declaration examples:
      - `String title;`
      - `String title = "Walls and Mirrors";`
    - Assignment example:
      - `Title = "Walls and Mirrors";`
    - String length example:
      - `title.length();`
    - Referencing a single character
      - `title.charAt(0);`
    - Comparing strings
      - `title.compareTo(string2);`

Useful Java Classes

- String classes (continued)
  - Class `String`
    - Concatenation example:

```
String monthName = "December";
int day = 31;
int year = 02;
String date = monthName + " " + day + ",
 20" + year;
```


Useful Java Classes

- String classes (continued)
  - Class `StringTokenizer`
    - Allows a program to break a string into pieces or tokens
    - More useful methods
      - `public StringTokenizer(String str)`
      - `public StringTokenizer(String str, String delim)`
      - `public StringTokenizer(String str, String delim, boolean returnTokens)`
      - `public String nextToken()`
      - `public boolean hasMoreTokens()`

Text Input and Output

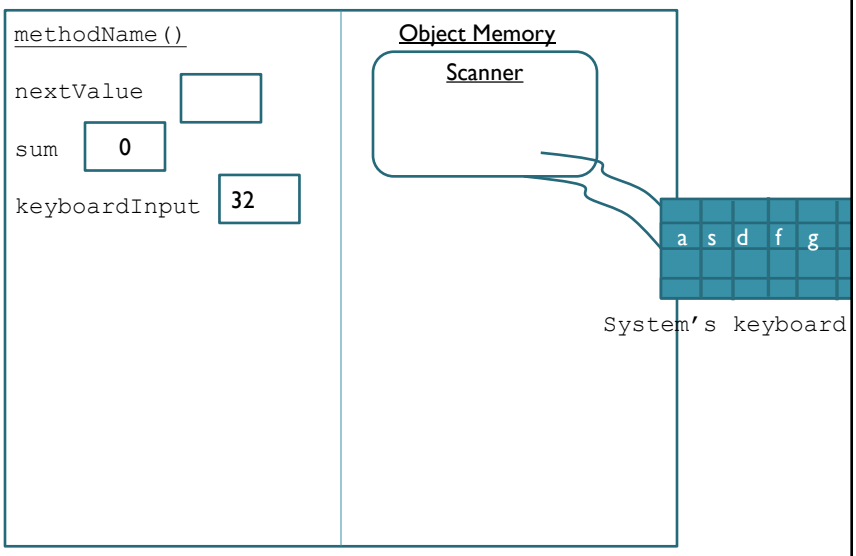
- Input and output consist of streams
- Streams
  - Sequence of characters that either come from or go to an I/O device
  - `InputStream` – Input stream class
  - `PrintStream` – Output stream class
- java.lang.System provides three stream variables
  - `System.in` – standard input stream
  - `System.out`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

 1-15

A View of Scanner's Execution



Input

More useful `Scanner` class methods

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

 1-17

<<<< It is expected that the rest of these slides will be used in Thursday's class >>>>

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

 1-20

Output

Method `printf`

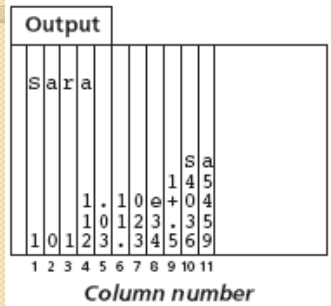
- C-style formatted output method

```
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
```

Multi Dimensional Arrays

- Use more than one index
- For Example

```
final int DAYS_PER_WEEK = 7;
final int WEEKS_PER_YEAR = 52;
double[][] minTemps = new
  double[DAYS_PER_WEEK][WEEKS_PER_YEAR];
```

Arrays

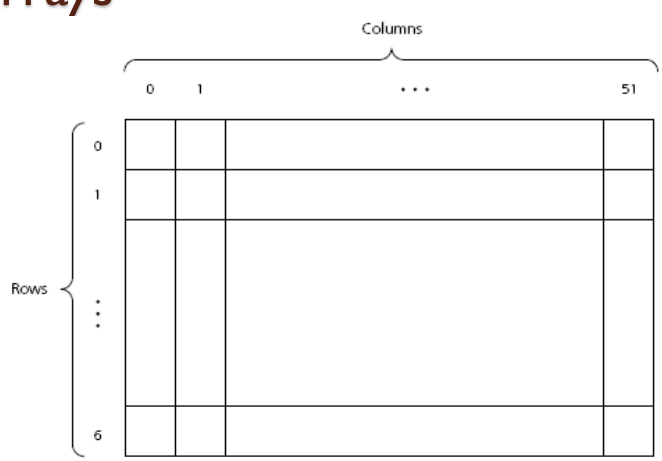


Figure 1-8
A two-dimensional array

Example II:
- Illustrates Arrays

- Write a program that:
  - inputs from a file that contains the following data: 5 temperatures per day for 30 days
  - calculates and outputs the average daily temperature and the daily minimum and maximum temperature
- And be able to pass that array to a method!

```
public class Temperatures {
  public static void main(String [] args) {

    // Memory space to store the array
    double[][] temperature = new
       double[DAYS][NUMBER_PER_DAY];

    //Inp
    Scann
     Strin                                    t";
     File

    for (
      for                                     ++) {
        //                                    e ==> ");
        te                                    le();
      }
    }
    //

  }
}
```

See Lecture 3 (or maybe 4) Code for program developed in class that is 1) correct and 2) does all the things specifed on the previous slide.

```
public class Temperatures {
  public static void main(String [] args) {

    // Memory space to store the array
    double[][] temperature = new
       double[DAYS][NUMBER_PER_DAY];

    //Inp
    Sc
     Sc
     Fl

      for(
       for
        f
        te

            maximum[day] = temperature[day][index];
          }
          // output result
          System.out.print("Day "+(day+1) +
                      ":Average"+sum/NUMBER_PER_DAY + " ");
  }
          System.out.println("Max " +maximum+ " Min "
```

See Lecture 3 (or maybe 4) Code for a readable version of the program developed in class.

Summary: Chapter 1: Java Background

- `import` statement
  - Required to use classes contained in other packages
- Object in Java is an instance of a class
- Class
  - Data type that specifies data and methods available
  - Data fields are either variables or constants
  - Methods implement object behavior
- Method parameters are passed by value

Summary: Chapter 1: Java Background

- Comments in Java
  - Comment lines
  - Multiple-line comments
- Java identifier
  - Sequence of letters, digits, underscores, and dollar signs
- Primitive data types categories
  - integer, character, floating point, and boolean
- Java reference
  - Used to locate an object

Summary: Chapter 1: Java Background

- Define named constant with `final` keyword
- Java uses short-circuit evaluation for logical and relational expressions
- Array
  - Collection of references that have the same data type
- Selection statements
  - `if` and `switch`
- Iteration statements
  - `while`, `for`, and `do`

Summary: Chapter 1: Java Background

- String
  - Sequence of characters
  - String classes: `String`, `StringBuffer`, `StringTokenizer`
- Files: accessed using `Scanner` class (streams)

