CS1150
Principles of Computer Science
Introduction (Part II)

Yanyan Zhuang

Department of Computer Science

http://www.cs.uccs.edu/~yzhuang

Review

- Terminology
  - Class
    - Every Java program must have at least 1 class (same name as Java file)
  - Method
    - Java programs contains a special method called “main”
      - public static void main (String[] args) {…..}
      - The “main” method is where a Java program starts execution
  - Statements
    - Statements represent actions
    - Statements must end with a ;

Review

- Terminology
  - Reserved words (keywords)
    - Words that have specific meaning/cannot be used for other purposes
    - public, class, byte, int, long, float, double, …
  - Blocks
    - Class block and method block, using {}
  - Comments
    - // This is a line of comment
    - /* This comment can span

      across several lines */

Review

- Identifier
  - A name chosen by the programmer for: classes, methods, **variables, and constants**

Overview

- Numerical Data Types
- Variables and constants
- Packages
- Data formatting
- Data casting

Numerical Data Types

Name	Range	Storage Size
`byte`	-2^7 to $2^7 - 1$ (-128 to 127)	8-bit signed
`short`	-2^{15} to $2^{15} - 1$ (-32768 to 32767)	16-bit signed
`int`	-2^{31} to $2^{31} - 1$ (-2147483648 to 2147483647)	32-bit signed
`long`	-2^{63} to $2^{63} - 1$ (i.e., -9223372036854775808 to 9223372036854775807)	64-bit signed
`float`	Negative range: -3.4028235E+38 to -1.4E-45 Positive range: 1.4E-45 to 3.4028235E+38	32-bit floating point
`double`	Negative range: -1.7976931348623157E+308 to -4.9E-324 Positive range: 4.9E-324 to 1.7976931348623157E+308	64-bit floating point

- Reading for primitive data types:
  - https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

Numeric Operators

Name	Meaning	Example	Result
+	Addition	34 + 1	35
-	Subtraction	34.0 – 0.1	33.9
*	Multiplication	300 * 30	9000
/	Division	1.0 / 2.0	0.5
%	Remainder	20 % 3	2

Integer Division

+, -, *, /, and %

5 / 2 yields an integer **2**

5.0 / 2 yields a double value **2.5**

5 % 2 yields 1 (the remainder of the division) – often called modular operation

ComputeExpression.java: example

Remainder Operator %

- Remainder is very useful in programming
  - For example, an even number % 2 is always 0 and an odd number % 2 is always 1
  - So you can use this property to determine whether a number is even or odd

System.out.println() prints only strings, but converts number to string to print

Java Identifiers

- An identifier
  - Example
    - Class/method/variable/constant name (can refer by name later)
  - **Rule**
    - A sequence of characters that consist of letters, digits, underscores (_), and dollar signs ($), NO SPACES
    - Must start with a letter, _ or $, CANNOT start with a digit
    - CANNOT be a reserved word, true, false, or null
    - Can be of any length
  - **Convention**
    - **Class** names start with an upper case letter, **variable/method** names start with a lower case letter, **constants** all caps

Variables and Constants

- Variable
  - Used to store a value that may change
  - E.g., double length; // **declare** a variable length

    length = 3.5; // **assign** 3.5 to length -- length can change

    double length = 3.5; // an (almost) equivalent way as above
  - How to use: Declare → assign a value → do something to it
    - A variable can only be declared **once** (double length;)
    - A variable must have a value (be initialized) before we use it (length=3.5;)
    - Can modify their value, display their value, use them in formulas
    - Example: Arithmetic.java, Circle.java (how to declare/use variables)

Variables and Constants

- Constant
  - Used to store a value that will **NEVER** change
  - Constants follow certain rules
    - Must have a name (a meaningful name, like variables)
      - With all uppercase letters (Java convention)
    - Declared using the keyword **final**
      - Example: final double PI = 3.14159;
    - Circle.java uses a constant

Write pseudocode

- Not real code, but help organize program logic
- Can use a mix of programming language + natural language
- Example: Calculate the area of a circle (Circle.java)

```
Input: radius
Output: area of the circle

area = pi * radius * radius
Print area
```

Also see Rectangle.java

Reading Input from the Console

1. Import java.util.Scanner and create a Scanner object

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

2. Use the method nextDouble() to obtain to a double value. For example,

```
System.out.print("Enter a double value:");
double d = keyboard.nextDouble();
```

3. After finishing the Scanner object, close it

```
keyboard.close();
```

Reading Input from the Console

```
Scanner keyboard = new Scanner(System.in);
int value = keyboard.nextInt();
```

Example: Average.java, ComputeArea1.java, ComputeArea2.java

Method	Description
`nextByte()`	reads an integer of the `byte` type.
`nextShort()`	reads an integer of the `short` type.
`nextInt()`	reads an integer of the `int` type.
`nextLong()`	reads an integer of the `long` type.
`nextFloat()`	reads a number of the `float` type.
`nextDouble()`	reads a number of the `double` type.

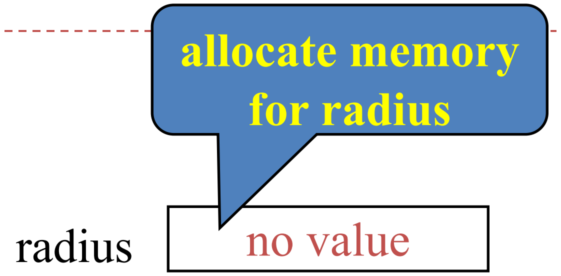
Trace a Program Execution

```
public class ComputeArea {
 /** Main method */
 public static void main(String[] args) {
   double radius;
   double area;

   // Assign a radius
   radius = 20.3;

   // Compute area
   area = radius * radius * 3.14159;

   // Display results
   System.out.println("The area for the circle of radius " +
     radius + " is " + area);
 }
}
```



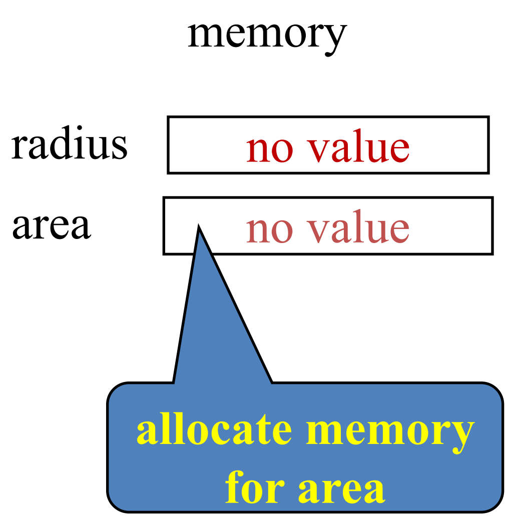
Trace a Program Execution

```
public class ComputeArea {
 /** Main method */
 public static void main(String[] args) {
   double radius;
   double area;

   // Assign a radius
   radius = 20.3;

   // Compute area
   area = radius * radius * 3.14159;

   // Display results
   System.out.println("The area for the circle of radius " +
     radius + " is " + area);
 }
}
```



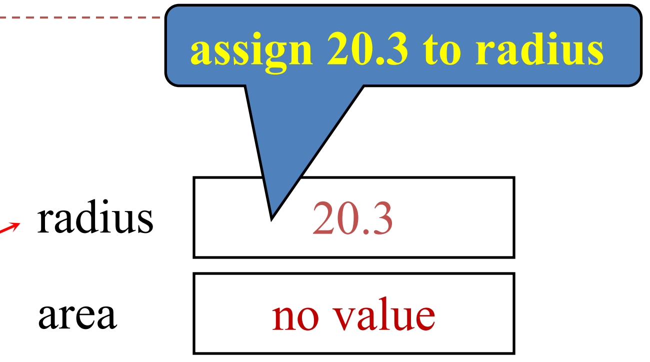
Trace a Program Execution

```
public class ComputeArea {
 /** Main method */
 public static void main(String[] args) {
   double radius;
   double area;

   // Assign a radius
   radius = 20.3;

   // Compute area
   area = radius * radius * 3.14159;

   // Display results
   System.out.println("The area for the circle of radius " +
     radius + " is " + area);
 }
}
```



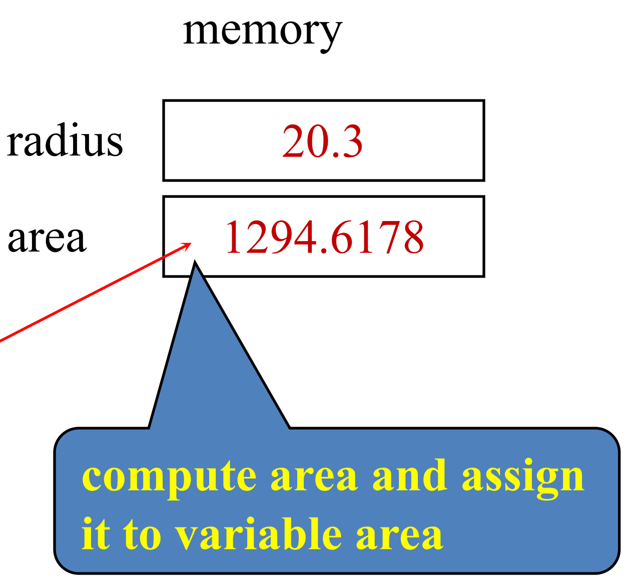
Trace a Program Execution

```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radius;
    double area;

    // Assign a radius
    radius = 20.3;

    // Compute area
    area = radius * radius * 3.14159;

    // Display results
    System.out.println("The area for the circle of radius " +
      radius + " is " + area);
  }
}
```



Trace a Program Execution

```
public class ComputeArea {
 /** Main method */
 public static void main(String[] args) {
  double radius;
  double area;

  // Assign a radius
  radius = 20.3;

  // Compute area
  area = radius * radius * 3.14159;

  // Display results
  System.out.println("The area for the circle of radius " +
   radius + " is " + area);
 }
}
```

memory

radius	20.3
area	1294.6178

print a message to the console

Packages

- A bunch of related classes grouped together
  - Mechanism for organizing Java classes
- Java contains many predefined packages
  - To access class/method in a predefined package use **import**
  - import java.util.Scanner;

Packages

- To create your own package
  - File (with project CS1150 selected) → New → Package
  - First line in your Java class: package your_package_name
- Structure
  - A project contains package(s)
  - A package contains Java file(s)
  - A Java file contains class(es)

Let’s Practice!

- Please write a program ComputeRadius.java: input the area of a circle, calculate its radius
  - Write pseudocode to guide your logic
  - Use constant PI
  - Use Scanner to get input value area (a double variable)
  - Print the value of the radius
    - radius = the square root of (area/PI)
    - How to do square root? Math.sqrt(area/PI) – you will use this in a homework question

Problem: Converting Temperatures

Write a program that converts a Fahrenheit degree (read from keyboard) to Celsius using the formula:

$$celsius = (\tfrac{5}{9})(fahrenheit - 32)$$

Print the resulting Celsius with **two decimal points**.

Converting Temperatures

- Pseudocode
  - Input? Output?
  - How to calculate Celsius given a Fahrenheit value?

$$celsius = (\tfrac{5}{9})(fahrenheit - 32)$$

Note: you have to write
celsius = (5.0 / 9) * (fahrenheit – 32)

  - How to print the result using two decimal points?
    - Data formatting

Formatting decimal output

- Use DecimalFormat class
  - import java.text.DecimalFormat;
  - DecimalFormat df = new DecimalFormat("000.##");
  - System.out.println(df.format(celsius));
  - 0: a digit
  - #: a digit, zero shown as absent
    - 72.5 is shown as 072.5
    - 21.6666….. is shown as 021.67
  - More information

    https://docs.oracle.com/javase/tutorial/i18n/format/decimalFormat.html

Formatting decimal output

- Use System.out.format
  - System.out.format("the %s jumped over the %s, %d times", "cow", ”moon", 2);
    - the cow jumped over the moon, 2 times
  - System.out.format("%.1f", 10.3456);
    - 10.3 // one decimal point
  - System.out.format("%.2f", 10.3456);
    - 10.35 // two decimal points
  - System.out.format("%8.2f", 10.3456);

    -
      ```
         10.35  // Eight-wide, two decimal points
      ```

  Example: Fahrenheit2Celsius.java

Let’s Practice!

- Please write a program: given the radius of a circle, calculate its perimeter, and print the result with 4 decimal points
  - Write pseudocode to guide your logic
  - Use constant PI
  - Use Scanner to get input value radius
  - Print the value of the perimeter

Data Casting

- When you **explicitly** tell Java to convert a variable from one data type to another type
  - Think of data types as bottles of different sizes
    - Can put the contents of a smaller variable (bottle) into a larger variable (bottle)
    - **Cannot** put the contents from a larger variable (bottle) into a smaller variable (bottle), **without losing information**
    - Cheat sheet: int (32 bits), long (64 bits), float (32 bits), double (64 bits)

Data Casting

- Convert a variable from one data type to another
  - Can put the contents of a smaller variable (bottle) into a larger variable (bottle)
    - double d = 3; // widening the type: 3 → 3.0
    - Java does this **automatically**
  - **Cannot** put the contents from a larger variable (bottle) into a smaller variable (bottle), **without losing information**
    - int num = (int)3.6; // narrowing the type: 3.6 → 3
    - Must be done **explicitly**
  - Cheat sheet: int (32 bits), long (64 bits), float (32 bits), double (64 bits)
  - Example: Casting.java

Conversion Rules

When performing a binary operation involving two operands of different types, Java **automatically** converts the operand based on the following rules:

The smaller type is converted to the larger type before operation occurs

1. If one of the operands is double, the other is converted into double.
2. Otherwise, if one of the operands is float, the other is converted into float.
3. Otherwise, if one of the operands is long, the other is converted into long.
4. Otherwise, both operands are converted into int.

Numeric Literals

- Variables and constants: have names
- Literals: A constant value that has no name
  - Integer literal
    - Values we assign to an integer variable: int i = **123**;
  - Floating-point literal
    - To indicate float/double, use suffix f/d
      - Leaving off the suffix, the number defaults to a **double**
    - Values assigned to float/double variable: float f = **12.34f**; or double d = **12.34**;
      - float floatValue = 71.71f; If leave off “f” would get error: cannot convert double to float
  - Scientific notation
    - 1.23456 x 10^{2} => 1.23456E2
    - 1.23456 X 10^{-2} => 1.23456e-2

Augmented Assignment Operators

- +, -, *, / and % operators
  - Each can be combined with the assignment operator (=)
    - a = a + 3; => a += 3;
    - Read the equation from **right to left**: variable a’s value plus 3, and assign it back to variable a
  - Same as -=, *=, /= and %=
    - a = a - 2; => a -= 2;
    - b = b*3.0; => b *= 3.0;
    - c = c % 5; => c %= 5;

Increment and Decrement Operators

- Increment: ++ Decrement: --
  - Operator can be placed before or after variables (postfix)

```
int i = 1, j = 3;
i++;    // Same as i = i + 1;   i will become 2
j--;    // Same as j = j – 1;   j will become 2
```

  - Alternatively (prefix)

```
int i = 1, j = 3;
++i;    // Same as i = i + 1;   i will become 2
--j;    // Same as j = j – 1;   j will become 2
```

  - Placement of prefix or postfix cause different results **when in expressions** so be careful (more details later)

Summary

- Data types and calculation
  - Variables and constants
- Reading input
- Data formatting/casting