204700
Data Structure and Programming Languages

Jakarin Chawachat

From: http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-092-introduction-to-programming-in-java-january-iap-2010/index.htm

2.MORE TYPES, METHODS, OPERATORS

Outline

- Lecture 1 Review
- More types
- Methods
- Conditionals

Types

Kind of values that can be stored and manipulated.

- **boolean**: Truth value (**true** or **false**).
- **int**: Integer (0, 1 -50)
- **double**: Real number (3.14, 1.0, -756.015)
- **String**: Text (“Hello world.”, “example”).

Variables

Named location that stores a value of one particular type.

Form:

TYPE NAME;

Example:

String foo;

Operators

Symbols that perform simple computations

Assignment: =

Addition: +

Subtraction: -

Multiplication: *

Division: /

Assignment 1

```
public class C2FConterter {
    public static void main(String[] args) {
        double celcius = 50;
        double farenheit = 0.0;
         farenheit = ((celcius/5) * 9) + 32;
        System.out.println(celcius+ "Celsius degree is equal to " + farenheit + "Fahrenheit degree.");
    }
}
```

Output of Assignment 1

50.0 Celsius degree is equal to 122.0 Fahrenheit degree.

Outline

- Lecture 1 Review
- More types
- Methods
- Conditionals

Division

Division (“/”) operates differently on integer and doubles!

Example:

```
double a = 5.0/2.0;          //  a = 2.5
int b = 4/2;                 //  b = 2
int c = 5/2;                 //  b = 2
double d = 5/2;              //  d = 2.0
```

Order of Operations

Precedence like math, left to right

Right hand side of = evaluated first

Parenthesis increase precedence

```
double x = 3/2+1;        //  x = 2.0;
Double y = 3/(2+1);      //  y = 1.0;
```

Mismatched Types

Java verifies that types always match:

```
String five = 5;    // ERROR!
```

```
Exception in thread "main" java.lang.Error:
    Unresolved compilation problem: Type
    mismatch: cannot convert from int to String
at C2FConterter.main(C2FConterter.java:6)
```

Conversion by casting

```
int a = 2;                      //  a = 2
double a = 2;                   //  a = 2.0 (Implicit)

int a= 18.7;                    // ERROR
int a= (int)18.7;               //18

double a =2/3;                  //  a = 0.0
double a = (double)2/3;         //  a = 0.6666….
```

Outline

- Lecture 1 Review
- More types
- Methods
- Conditionals

Methods

- A Java method is a collection of statements that are grouped together to perform an operation.

- When you call the System.out.println method, for example, the system actually executes several statements in order to display a message on the console.

Adding Methods

In general, a method has the following syntax:

```
modifier returnValueType methodName(list of
    parameters) {
        // Method body;
}
```

The parts of a method

- **Modifiers:** (optional) The modifier tells the compiler how to call the method. This defines the access type of the method.

- **Return Type:** The returnValueType is the data type of the value the method returns. (**void if no return value**)

- **Method Name:** This is the actual name of the method.

- **Parameters:** (optional) When a method is invoked, you pass a value to the parameter.

- **Method Body:** The method body contains a collection of statements that define what the method does.

Examples

Example: no return value

```
public static void threeLines() {
        System.out.println("");
    }
```

Examples

Example: return value

```
public static double pivalue() {
        return 3.14;
   }
```

Examples

Example: return value / parameter

```
public static double addition(int a, int b) {
        int result;
        result = a + b;
        return result;
    }
```

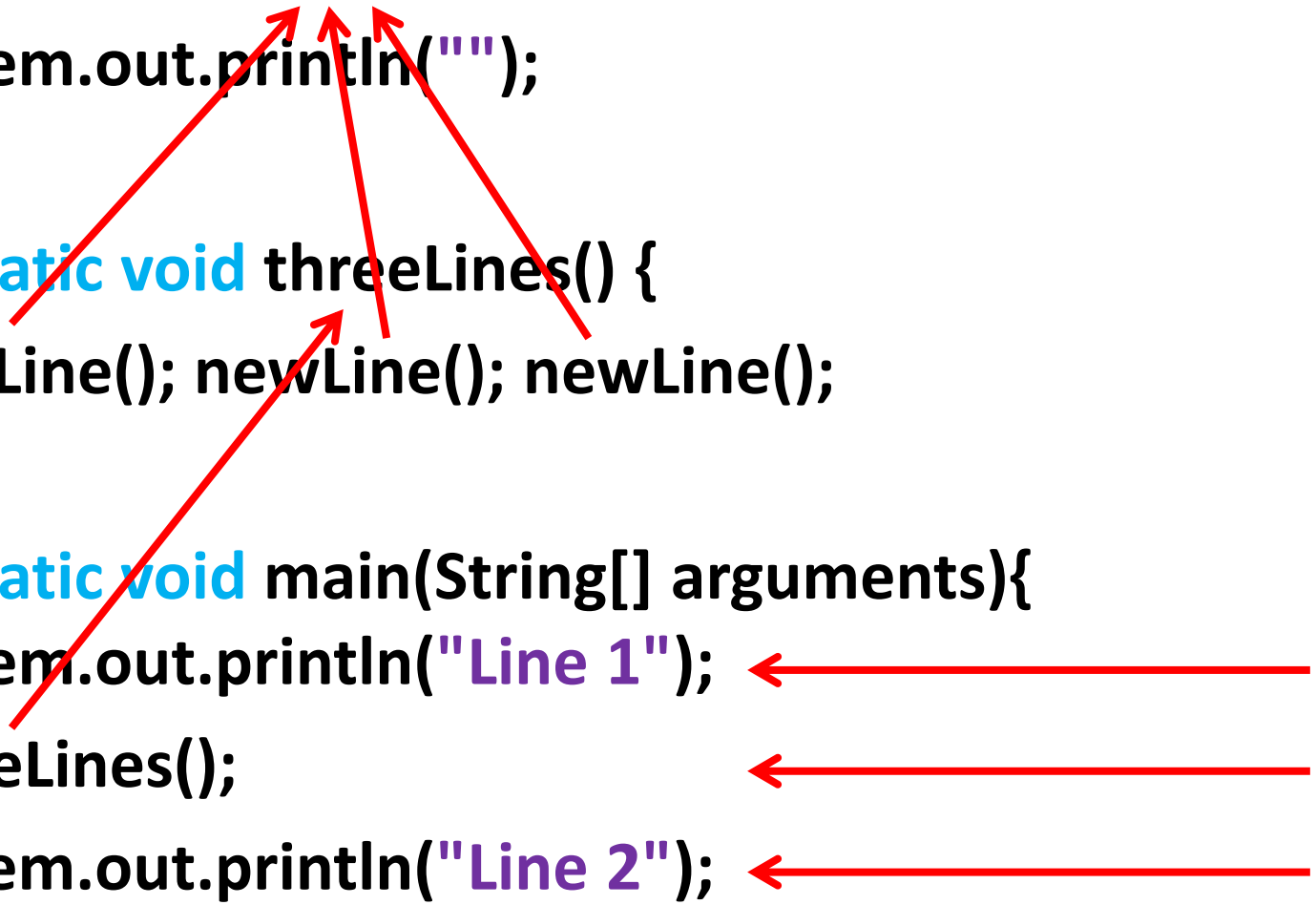
Adding Methods

```
public static void NAME(){
    STATEMENTS
}
```

To call a method:

```
NAME();
```

```
class NewLine {
    public static void newLine() {
        System.out.println("");
    }
    public static void threeLines() {
        newLine(); newLine(); newLine();
    }
    public static void main(String[] arguments){
        System.out.println("Line 1");
        threeLines();
        System.out.println("Line 2");
    }
}
```



Parameters

public static void ***NAME***(*TYPE NAME*){

    STATEMENTS

}

To call a method:

NAME(*EXPRESSION*);

```
class Square {
    public static void printSquare(int x){
        System.out.println(x * x);
    }
    public static void main(String[] arguments){
        int value = 2;
        printSquare(value);
        printSquare(3);
        printSquare(value * 2);
    }
}
```

```
class Square2 {
    public static void printSquare(int x){
        System.out.println(x * x);
    }
    public static void main(String[] arguments){
        printSquare("hello");
        printSquare(5.5);
    }
}
```

What's wrong here?

Multiple Parameters

public static void ***NAME***(*TYPE NAME, TYPE NAME*){

STATEMENTS

}

To call a method:

NAME(*arg1, arg2*);

```
class Multiply {
    public static void times (double a, double b){
        System.out.println(a * b);
    }
    public static void main(String[] arguments){
        times (2, 2);
        times (3, 4);
    }
}
```

Return Values

```
public static TYPE NAME(){
    STATEMENTS;
    return EXPRESSION;
}
```

void means “no type”

```
class Square3 {
    public static void printSquare(double x){
        System.out.println(x * x);
    }
    public static void main(String[] arguments){
        printSquare(5);
    }
}
```

```
class Square4 {
    public static double square(double x){
        return x*x;
    }
    public static void main(String[] arguments){
        double ans;
        ans = square(5);
        System.out.println(ans);
        System.out.println(square(2));
    }
}
```

Variable Scope

Variables live in the block ({}) where they are defined (scope)

Method parameters are like defining a new variable in the method

```
class SquareChange {
    public static void printSquare(int x){
        System.out.println("printSquare x = " + x);
        x = x * x;
        System.out.println("printSquare x = " + x);
    }
    public static void main(String[] arguments){
        int x = 5;
        System.out.println("main x = " + x);
        printSquare(x);
        System.out.println("main x = " + x);
    }
}
```

```
class Scope {
    public static void main(String[] arguments){
        int x = 5;
        if (x == 5){
            int x = 6;
            int y = 72;
            System.out.println("x = " + x + " y = " + y);
        }
        System.out.println("x = " + x + " y = " + y);
    }
}
```

Duplicate local variable x

Methods: Building Blocks

- Big programs are built out of small methods
- Methods can be individually developed, tested and reused
- User of method does not need to know how to it workd
- In Computer Science, this is called “abstraction”

Mathematical Functions

Math.sin(x)

Math.cos(Math.PI/2)

Math.pow(2, 3)

Math.log(Math.log(x+y))

Outline

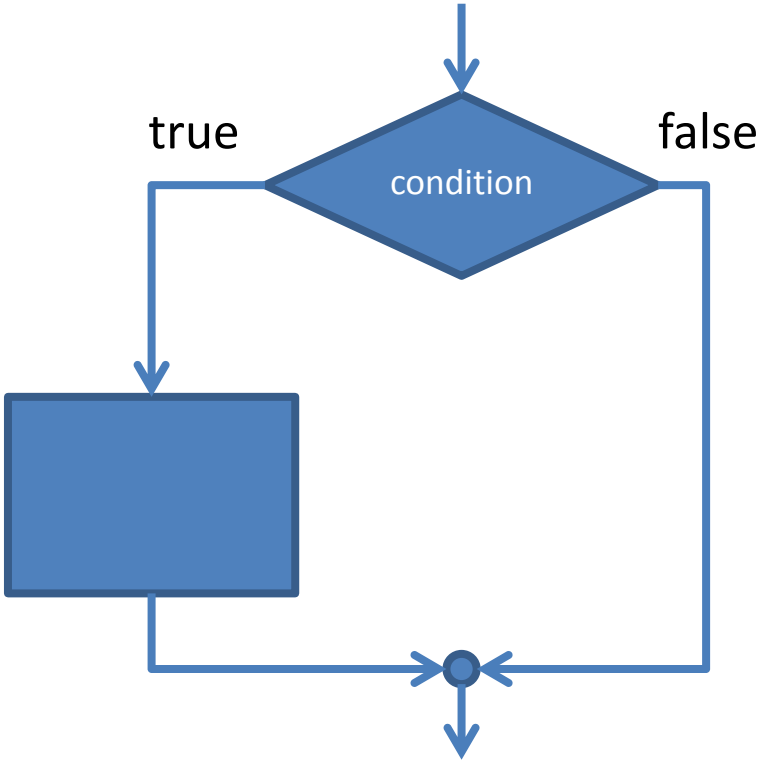
- Lecture 1 Review
- More types
- Methods
- Conditionals

If statement

if (***CONDITION***) {

    STATEMENTS

}



```
public static void test(int x){
    if (x > 5){
        System.out.println(x + " is > 5");
    }
}
public static void main(String[] arguments){
    test(6);
    test(5);
    test(4);
}
```

Comparison operators

x > y: x is greater than y

x < y: x is less than y

x >= y: x is greater than or equal to y

x <= y: x is less than or equal to y

x == y: x equals to y

(equality: ==, assignment: =)

Boolean operators

&&: logical AND

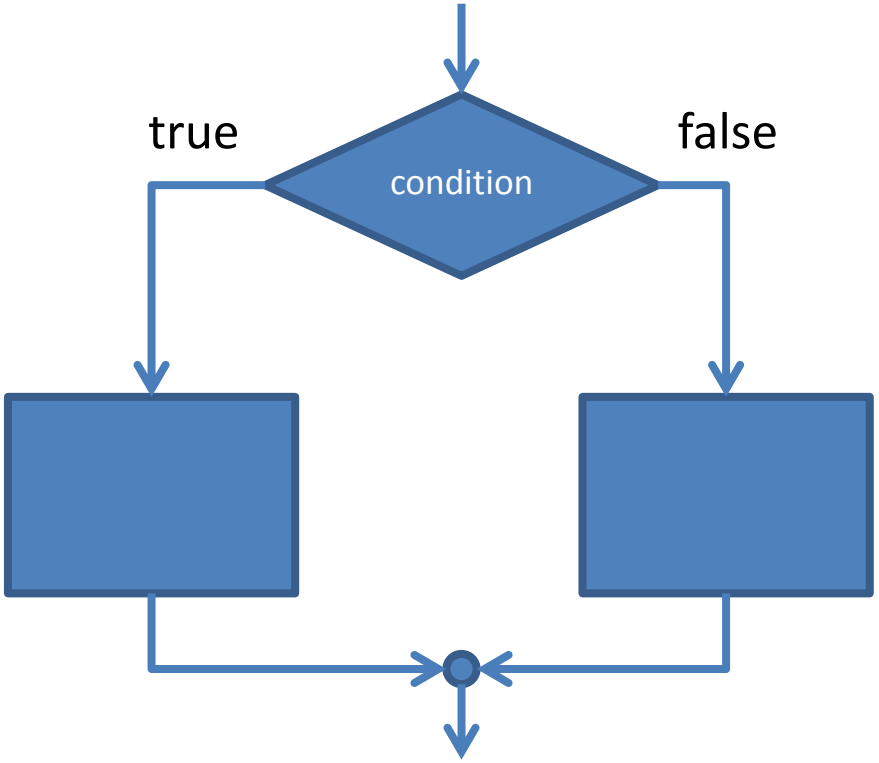
||: logical OR

```
if(x > 6){
    if(x < 9){
            ....
    }
}
```

```
if (x > 6 && x < 9) {
            …
}
```

else

```
if (CONDITION) {
    STATEMENTS
} else {
    STATEMENTS
}
```

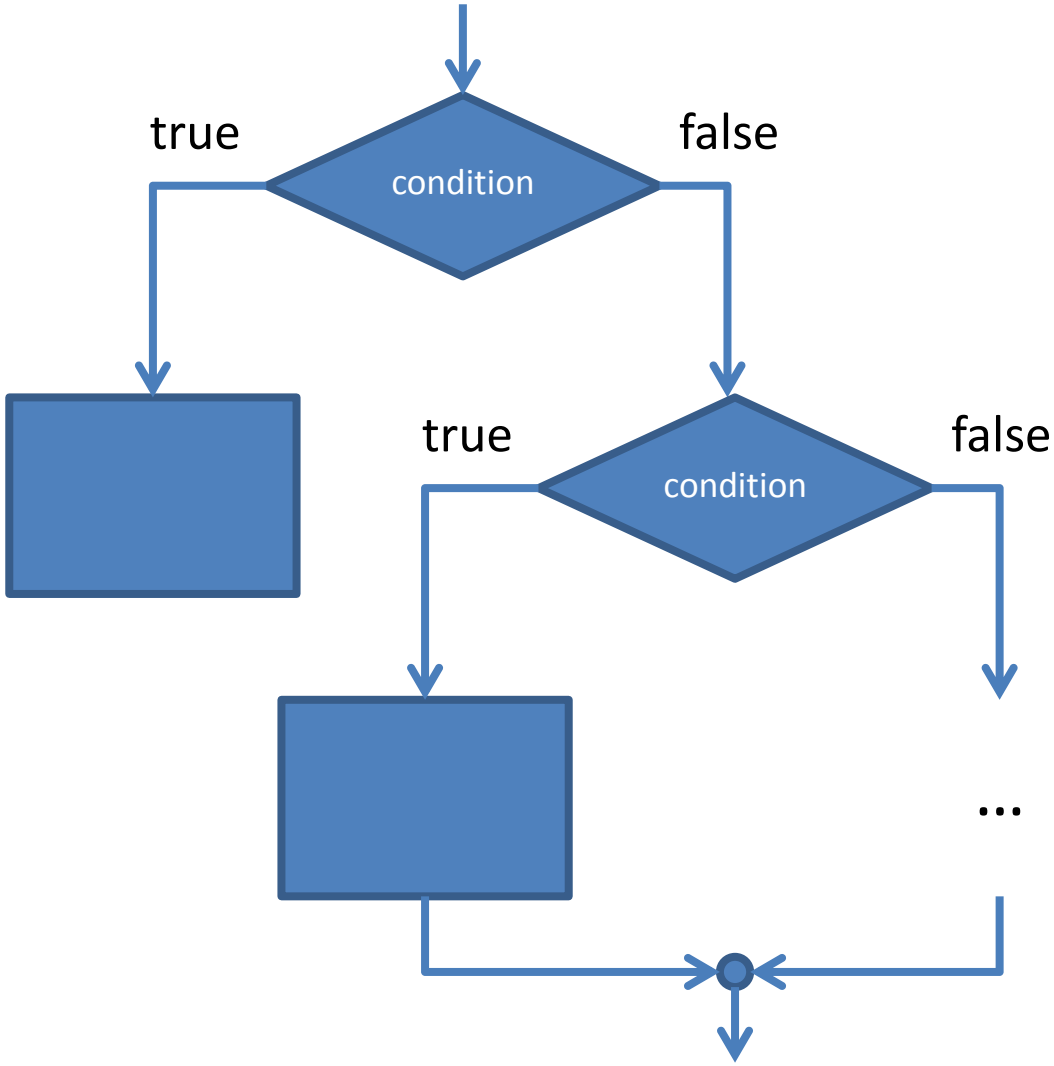


```
public static void test(int x){
    if (x > 5){
        System.out.println(x + " is > 5");
    } else {
        System.out.println(x + " is not > 5");
    }
}
public static void main(String[] arguments){
    test(6);
    test(5);
    test(4);
}
```

```
6 is > 5
5 is not > 5
4 is not > 5
```

Else if

```
if (CONDITION) {
    STATEMENTS
} else if (CONDITION) {
    STATEMENTS
} else if (CONDITION) {
    STATEMENTS
} else }
    STATEMENTS
}
```



```
public static void test(int x){
    if (x > 5){
        System.out.println(x + " is > 5");
    } else if (x == 5){
        System.out.println(x + " equals 5");
    } else {
        System.out.println(x + " is < 5");
    }
}
public static void main(String[] arguments){
    test(6);
    test(5);
    test(4);
}
```

```
6 is > 5
5 equals 5
4 is < 5
```

Conversion be method

- int to String:
  - String five = 5; //ERROR
  - String five = Integer.toString(5);
  - String five = “” + 5; //five = “5”
- String to int:
  - int foo = “18”; //ERROR!
  - Int foo = Integer.parseInt(“18”);

Comparison operators

- Do not call == on doubles! EVER.

```
double a = Math.cos(Math.PI/2);
double b = 0.0;
```

a = 6.123233995736766E-17

a = b will return FALSE;

Assignment 2

Method to print pay based on base pay and hour worked

Overtime: More than 40 hours, paid 1.5 times base pay

- Minimum Wage: $8.00/hour
- Maximum Work: 60 hours a week