

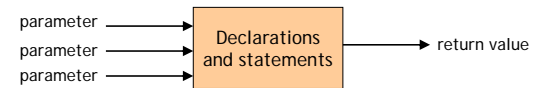
Introduction to Programming

with Java, for Beginners

Static Methods

About methods

- A **method** is a named group of declarations and statements
 - You execute those declarations and statements by **calling** the method
 - When you call the method, you can give it **parameters** (information, such as numbers, going into the method)
 - A method typically has a **return value** (a single piece of information coming out of the method)



ESE112

1/13

Why methods?

1. Modularity

- Break up a complex problem into simpler sub-problems, which you can solve separately
- Note: Methods can use other methods, but should not depend on the “inner workings” of those other methods--just what they do, not how they do it

2. Write once and reuse

- This is an application of the **DRY** principle (“Don’t Repeat Yourself”)

ESE112

2/13

Method Names

- The reusable parts of your computation can be given a name
- Proper choice of method names is important
 - Verbs are usually best, since methods “do something”
 - Naming rules are the same for naming variables
 - Descriptive names make your program more readable

ESE112

3/13

Method Syntax

- A static method has the syntax:

```
static return-type method-name(parameters) {  
    method-variables  
    code  
}
```

- Example:

```
static boolean isAdult(int age) {  
    int magicAge = 21;  
    return age >= magicAge;  
}
```

- Example:

```
static double average(int a, int b) {  
    return (a + b) / 2.0;  
}
```

ESE112

4/13

Returning a result from a method

- If a method is to return a result, it must specify the *type* of the result:

- static boolean isAdult (...

- You must use a return statement to exit the method with a result of the correct type:

- return age >= magicAge;

ESE112

5/13

Returning *no* result from a method

- The keyword **void** is used to indicate that a method doesn't return a value

- The **return** statement must not specify a value

- Example:

```
static void printAge(String name, int age) {  
    System.out.println(name + " is " + age + " years old.");  
    return;  
}
```

- There are two ways to return from a void method:

- Execute a return statement
- Reach the closing brace of the method (so you can avoid the return statement altogether)

ESE112

6/13

Circle.java

```
class Circle{  
    static double area (double radius) {  
        final double PI = 3.14;  
        return radius * radius * PI;  
    }  
}
```

ESE112

7/13

Accessibility Level

- Class methods and class itself can also have **accessibility_level**
- Examples: **public** static double area (double radius) { .. }
- **public**: makes the method accessible from outside the class
- **Private**: not accessible outside the class
- **In default case** (i.e. no mention of public and private): accessible if within same directory
- Accessibility_level *appears before*
 - *static* keyword for static method
 - *class* keyword for class description

ESE112

8/13

Java Classes and static Methods

- Methods are always written within a class
 - To execute the method declarations and statements we **call** the method
- Two ways to call a method:
 1. Within the same class:
staticMethodName(parameters);
E.g. double a = area(3.0);
 2. Outside class in which it is declared in:
Classname.staticMethodName(parameters);
E.g. double a = Circle.area(3.0);

ESE112

9/13

Main Method

static void main(String [] args)

- A special static method
 - Whose return type is **void** and
 - Input is a **String array** ([]) (more or arrays later)
- Entry point of a java program i.e. where the instructions starts to get executed step by step
 - If there is variable declared, then space is allocated in memory
 - If it comes across method call, then method declaration and statements are executed
 - Until the last statement, after which terminates the program

ESE112

10/13

Option 1 w/ Main Method

```
public class Circle {  
    public static void main(String [] args) {  
        double a = area(3.0);  
        System.out.println("Area = " + a);  
    }  
    public static double area (double radius) {  
        final double PI = 3.14;  
        return radius * radius * PI;  
    }  
} //end of Circle class
```

ESE112

11/13

Option 2 w/ Main Method

```
public class TestCircle {
    public static void main(String [] args) {
        double a = Circle.area(3.0);
        System.out.println("Area = " + a);
    }
} // end of TestCircle class

public class Circle {
    public static double area (double radius) {
        final double PI = 3.14;
        return radius * radius * PI;
    }
} //end of Circle class
```

ESE112

12/13

Non-static methods

- This slide set only talks about static methods
- Most methods in a Java program will *not* be static
- We have to introduce classes and methods so that we can start writing some programs
- So, more to come...

ESE112

13/13