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)

### 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”)

### 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
Method Syntax

- A static method has the syntax:
  ```java
  static return-type  method-name (parameters) {
    method-variables
    code
  }
  ```

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

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

Returning a result from a method

- If a method is to return a result, it must specify the type of the result:
  ```java
  static boolean isAdult ( ... }
  ```

- You must use a return statement to exit the method with a result of the correct type:
  ```java
  return age >= magicAge;
  ```

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:
  ```java
  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)

Circle.java

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

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);

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

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
Option 2 w/ Main Method

```java
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
```

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...