Unit 2: Programming

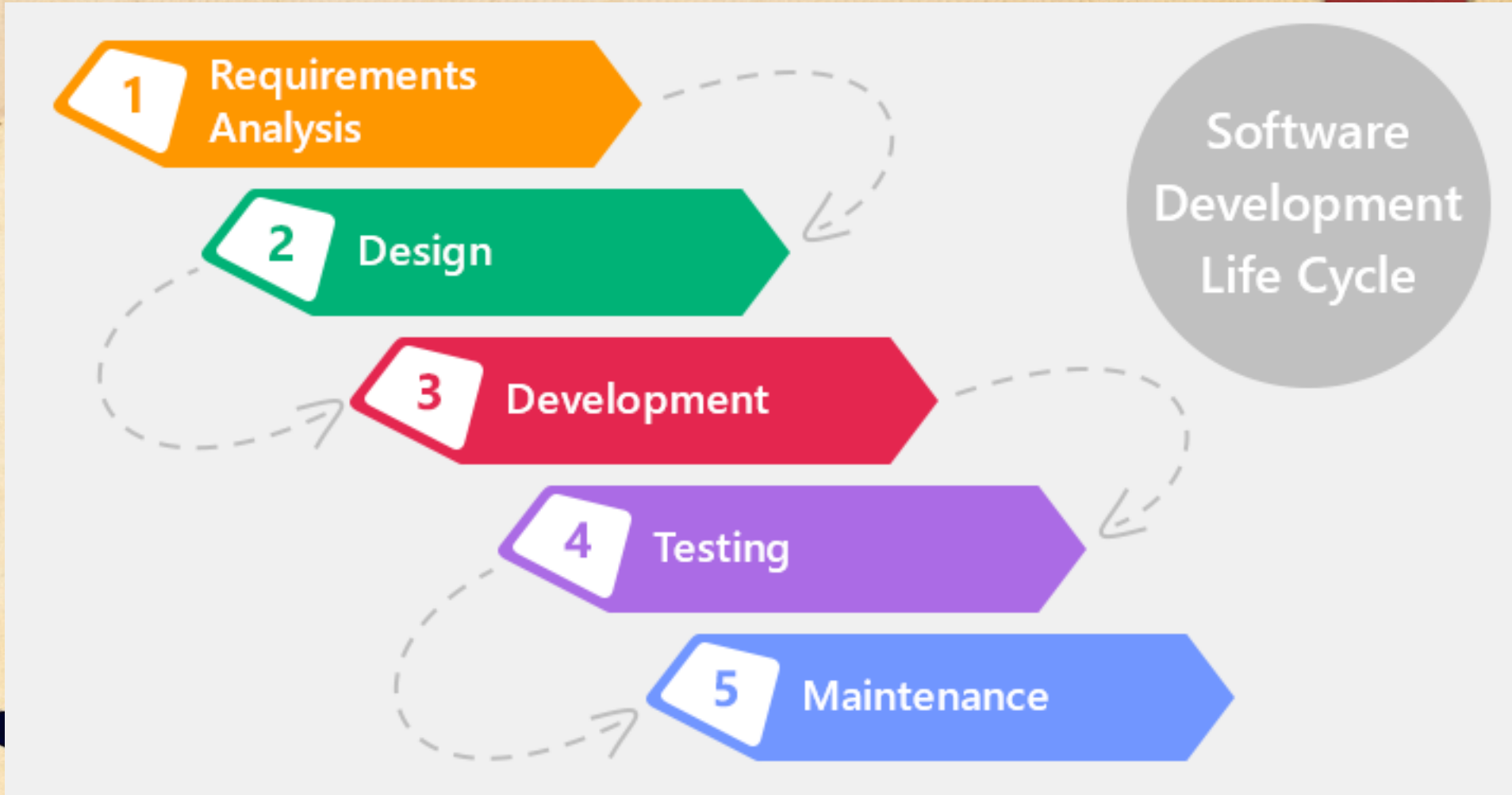
Topic 6: Making Programs Easy to Read

Lecture Contents

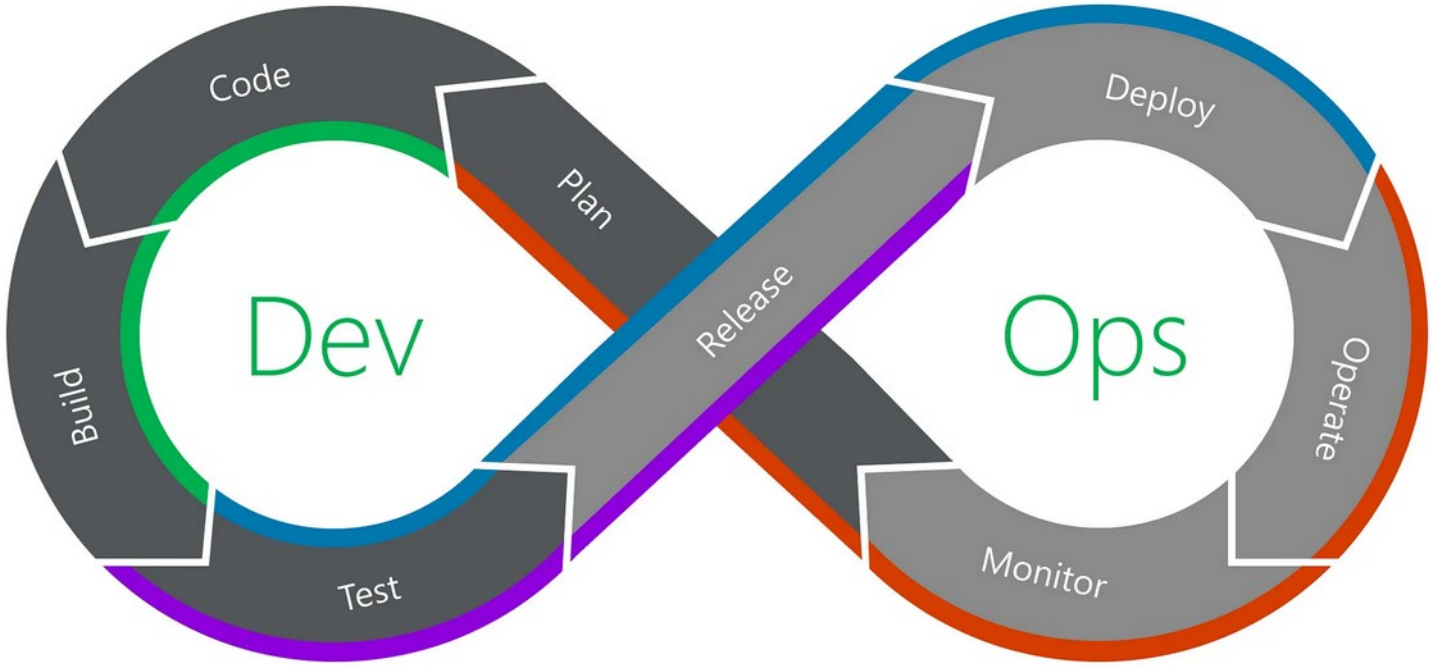
- Software Development Life Cycle
- Purpose of Comments
- Types of Comments in *Java*
- *Javadoc*
- Comment Requirements for Our Class

Making Programs Easy to Read

- Code should be written in standard ways
  - ***Naming conventions***
  - ***Indentation*** and ***white space***
  - ***Comments*** must aid in understanding and readability
- Standards do vary between languages, and even companies


Software Development Life Cycle
1 Requirements Analysis
2 Design
3 Development
4 Testing
5 Maintenance

Communication, Collaboration and Security



Continuous Integration (CI)

Continuous Deployment (CD)

Continuous Delivery (CD)

Continuous Feedback (CF)

Purpose of Comments

- Comments are NOT executed
- Comments are text added to the *source code* to provide explanatory information
- Why? Well, for **maintenance**.
  - Corrective maintenance
  - Adaptive maintenance
  -
- Can also be used to temporarily prevent execution of code

Types of Comments in *Java*

- Single-line comments
  - Text after a double forward slash (`//`) until the end of the line
- Multi-line comments
  - Any line between `/*` and `*/`
- Documentation comments
  - Any line between `/**` and `*/`
  - It is used for automatically generated documentation

`javadoc`

- A tool that comes with JDK
- Generates documentation in HTML format from Java source code documentation comments
- The documentation forms the API specification of the class
  - Allowing other programmers to use the class without reading the code and understanding the implementation
- Online Java documentation has been created from the `javadoc` comments in the Java libraries.

docs.oracle.com

String (Java Platform SE 8)

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

Java™ Platform Standard Ed. 8

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

compact1, compact2, compact3

java.lang

Class String

java.lang.Object
    java.lang.String

All Implemented Interfaces:

Serializable, CharSequence, Comparable<String>

```
public final class String
extends Object
implements Serializable, Comparable<String>, C
```

The String class represents character strings. All stri
such as "abc", are implemented as instances of this cl

Strings are constant; their values cannot be changed a
buffers support mutable strings. Because String objec
shared. For example:

```
String str = "abc";
```

The Java documentation online is generated from the library classes

```
/**
 * The {@code String} class represents character string
 * string literals in Java programs, such as {@code "al
 * implemented as instances of this class.
 * <p>
 * Strings are constant; their values cannot be changed
 * are created. String buffers support mutable strings
 * Because String objects are immutable they can be sha
 * <blockquote><pre>
 *     String str = "abc";
 * </pre></blockquote><p>
 * is equivalent to:
 * <blockquote><pre>
 *     char data[] = {'a', 'b', 'c'};
 *     String str = new String(data);
 * </pre></blockquote><p>
```

Comment Requirements for Our Class

- Above Class: Javadoc Comment
  - Brief description
  - @author
- Above Method: Javadoc Comment
  - Brief description
  - @param
  - @return
- Within Methods: Comments
  - Single- or multi-line comments explaining any code that is not obvious

Indentation and Spacing

- Indentation shows ***scope***
  - For Java, indent blocks of code
    - Inside methods, if statements, while statements, for loops, …
    - Generally, within curly braces { … }

```
public static void main(String[] args) {
   for( int I = 0; I < 10; i++) {
      System.out.print(i + " ");
      if( i%2 == 0) {
         System.out.println();
      }
   }
}
```

Making Programs Easy to Read

- Code should be written in standard ways
  - ***Naming conventions***
  - ***Indentation*** and ***white space***
  - ***Comments*** must aid in understanding and readability
- Standards do vary between languages, and even companies

Types of Comments in *Java*

- Single-line comments
  - Text after a double forward slash (`//`) until the end of the line
- Multi-line comments
  - Any line between `/*` and `*/`
- Documentation comments
  - Any line between `/**` and `*/`
  - It is used for automatically generated documentation

Unit 2: Programming

Topic 6: Making Programs Easy to Read