CMSC 131
Object-Oriented Programming I

Design

Dept of Computer Science
University of Maryland College Park

This material is based on material provided by Ben Bederson, Bonnie Dorr, Fawzi Emad, David Mount, Jan Plane

Overview

- Flags Project
- Design

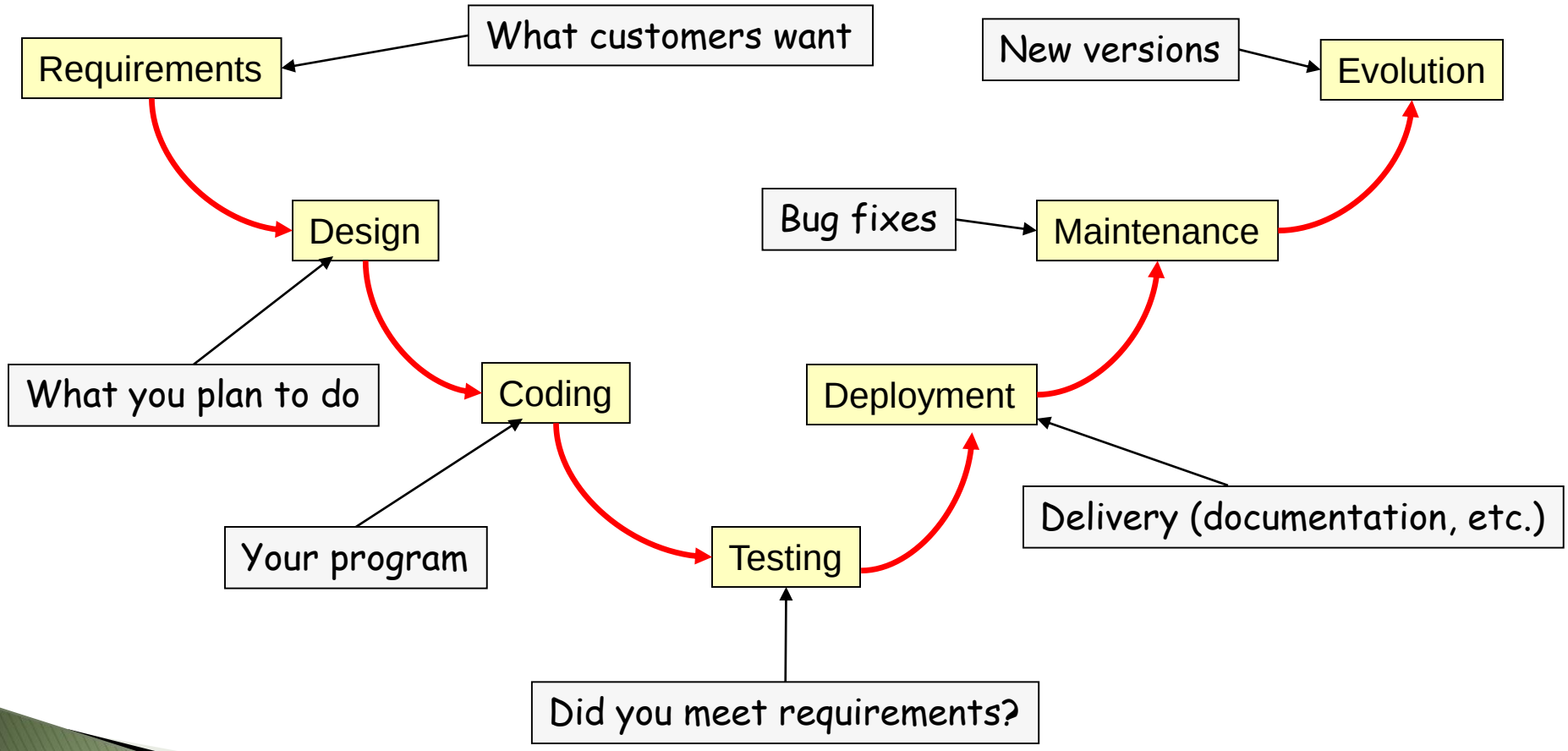
Flags Project

- Let's talk about the Flags Project
- Example (our FlagMakerLecture.java)
  - Shows the methods you need
  - Notice the coordinate system

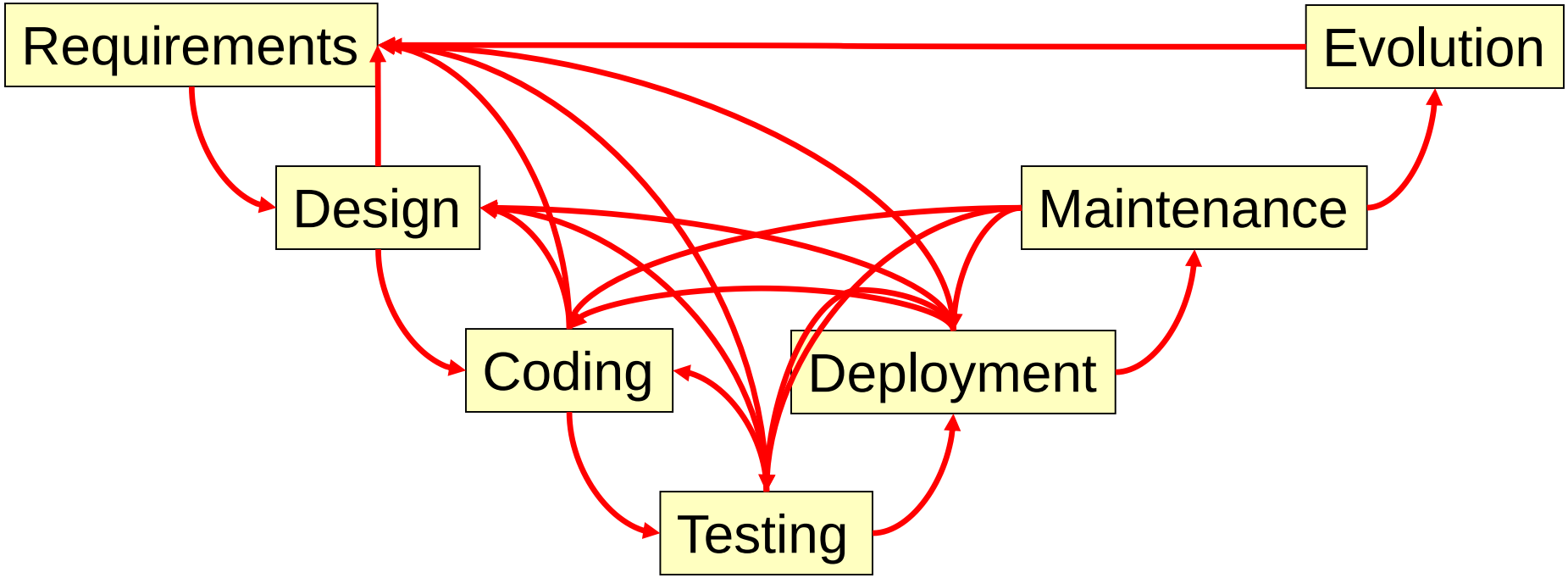
Java API

- Provides information about Java Classes/Interfaces
- http://docs.oracle.com/javase/8/docs/api/index.html
- Check the entry for the **Color** class

The Software Lifecycle ("waterfall")



The Software Lifecycle (actual)



In the Real World, Requirements and Design Rule

- Getting requirements right is essential for successful projects
  - FBI electronic case file (junked after $180m)
  - IRS system upgrade in late 90s (junked after >$2bn)
  - FAA air-traffic control (false starts, >$10bn spent)
- Good design makes other parts of lifecycle easier
- In “the real world” coding typically < 30% of total project costs
- A good design improves:
  - efficiency (speed)
  - efficiency (memory)
  - ease of coding
  - ease of debugging
  - ease of expansion

Program Design

- There are many aspects to good design
  - Architecture
  - Modeling
  - Requirements decomposition
  - Pseudo-code
- In this class we will focus on latter

Designing Using Pseudocode

- So far we have focused on the syntax and semantics of Java
- As the complexity of problems increase you need a design strategy to solve such problems
- Several alternatives exist to come up with a solution to a problem. A popular one is Pseudocode

- ***Pseudocode:*** *English-like description of the set of steps required to solve a problem*

- When you write pseudocode you focus on determining the steps necessary to solve a problem without worrying about programming language syntax issues

Pseudocode Example

Pseudocode for finding the minimum value

1. Read number of values to process (call this value n)
2. Repeat the following steps until the n input values have been processed
    a. Read next value into x
    b. If (x is the first value read) {
        currentMinimum = x
       } else {
        if (x < currentMinimum)
            currentMinimum = x
       }
3. Print currentMinimum value

Pseudocode Elements

- When writing pseudocode you need the following constructs:
  - Input
  - Output
  - Assignments
  - Repetition Structures
  - Conditionals
- To help you with the design of pseudocode you can use the following syntax to represent the above constructs

Pseudocode Elements

- **Input**
  - variable = read() e.g., x = read()
- **Output**
  - print(variable) e.g., print(x)
- **Assignment**
  - x = <value> e.g., x = 20, s = “Bob”
- **Repetition**

```
while (expression) {
    stmts
}
```

OR

```
do {
    stmts
} while (expression)
```

Pseudocode Elements

Conditional (1)

```
if (expression) {
   stmts
}
```

Conditional(2)

```
if (expression) {
   stmts
} else {
   stmts
}
```

Conditional (3)

```
if (expression1) {
   stmts
} else if (expression2) {
   stmts
   …
} else if (expressionN) {
   stmts
} else {
   stmts
}
```

How Good is Your Pseudocode

- Your code does not use language constructs that are particular to a programming language
- Anyone receiving the pseudocode will not need to ask you questions in order to transform the pseudocode into code (no matter what the target programming language is)

Algorithms are Important

- Dijkstra's Algorithm for Shortest Path
  - We may not be able to come up with it, but we can implement it
  - Dijkstra designed it in 20 minutes without pencil and paper
- Dijkstra is a Turing Award Winner
- Turing Award
  - http://awards.acm.org/homepage.cfm?srt=all&awd=140