Object Oriented Analysis and Design
CPE 343

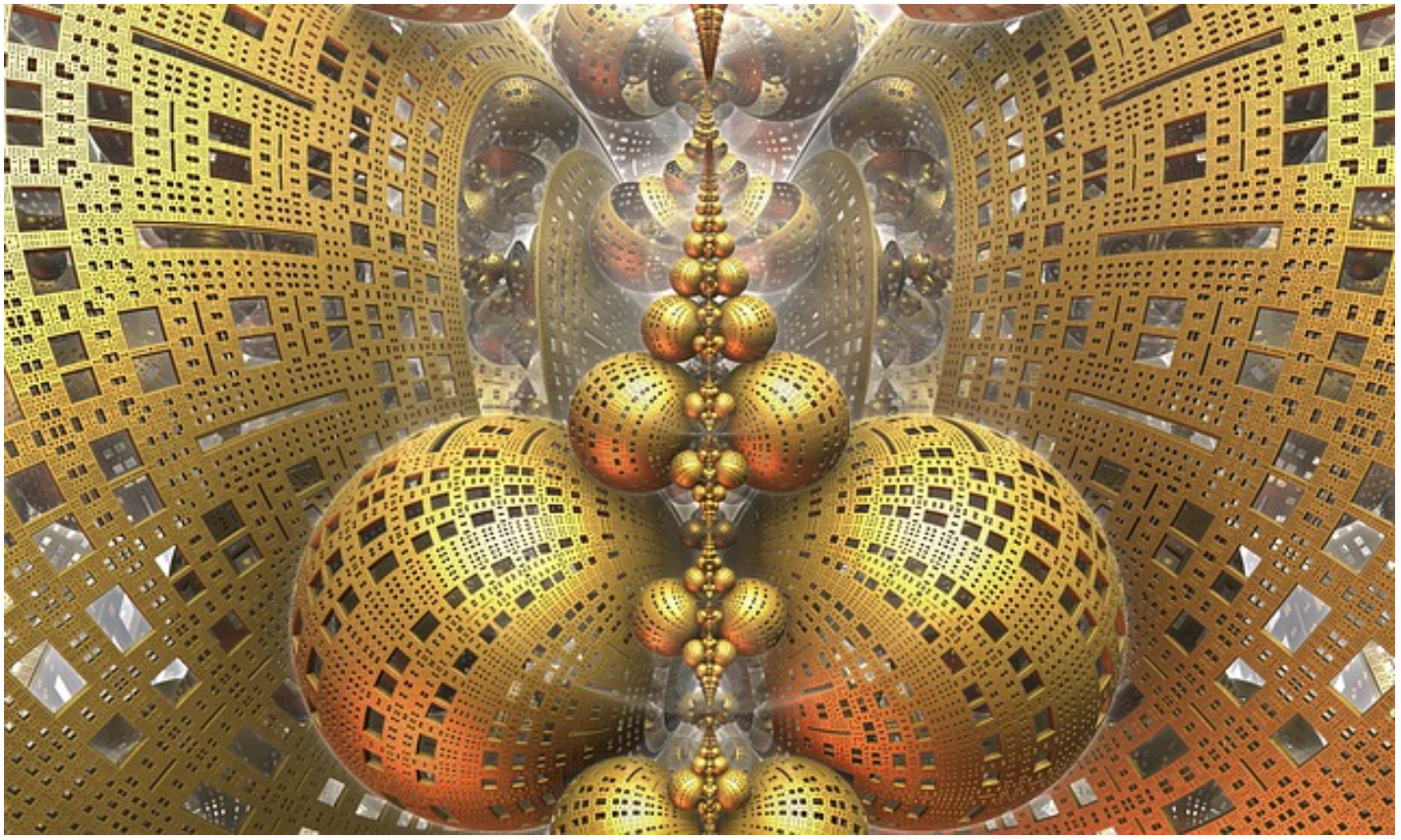
Lecture 1

What is OOD? Why use it?

Dr. Sally E. Goldin
Department of Computer Engineering
King Mongkut's University of Technology Thonburi
Bangkok, Thailand

Last update: 19 Jan 2020

Problem: Software Complexity



Hundreds of features
Thousands of functions
Tens of thousands of lines of code
Capabilities distributed across many different computing nodes

Problem: Teams and Time

Modern software is created by developer teams, not individual programmers
Software lifetimes measured in years, with constant change
Team members come and go – the software must continue to function and grow

Software Engineering Principles

- Software engineering (SE) tries to solve these problems
- Goal is to impose discipline & consistency on all aspects of development
- SE includes principles, processes & tools
- Important principles
  - Modularity: break up functionality into well-defined pieces; design, develop, test & maintain independently
  - Low coupling: minimize dependencies between modules
  - Abstraction: focus on and expose behavior not implementation details

Modularity in Procedural Languages

- C is a *procedural* language
- Modularity supported by:
  - Functions
  - Source files
  - Libraries
- Procedural languages generally do not *enforce* modularity
- Applying SE principles depends on discipline and skill of each developer

Object oriented languages enforce modularity!

Object oriented design (OOD) is a SE process intended to maximize modularity and abstraction in software systems

Modules in OO Languages

- Modules in an OO language are called *objects*
- An object combines data and behavior in a single package
- Modules interact with one another by requesting behaviors
- Can be viewed as one object *sending messages* to another
- Often modules represent the “natural” concepts in an application domain

Example: Banking System

- **Procedural view**
  - Open account
  - Deposit
  - Withdraw
  - Calculate fees
  - Compute balance
  - Close account
  - Display statement

- **Object-oriented view**
  - Account
    - *Open, close, get details*
  - Deposit
    - *Apply, cancel*
  - Withdrawal
    - *Apply, cancel*
  - Fee
    - *Calculate, apply*
  - Balance
    - *Calculate, display*
  - Statement
    - *Display, save, print*

Example: Email Client

- **Procedural view**
  - Get messages
  - Read message
  - Delete message
  - Compose message
  - Send message
  - Create folder
  - Move message to folder
  - Search by ‘From:’ address
  - Search by ‘To:’ address
  - Etc.
- **Object-oriented view**

  ???????

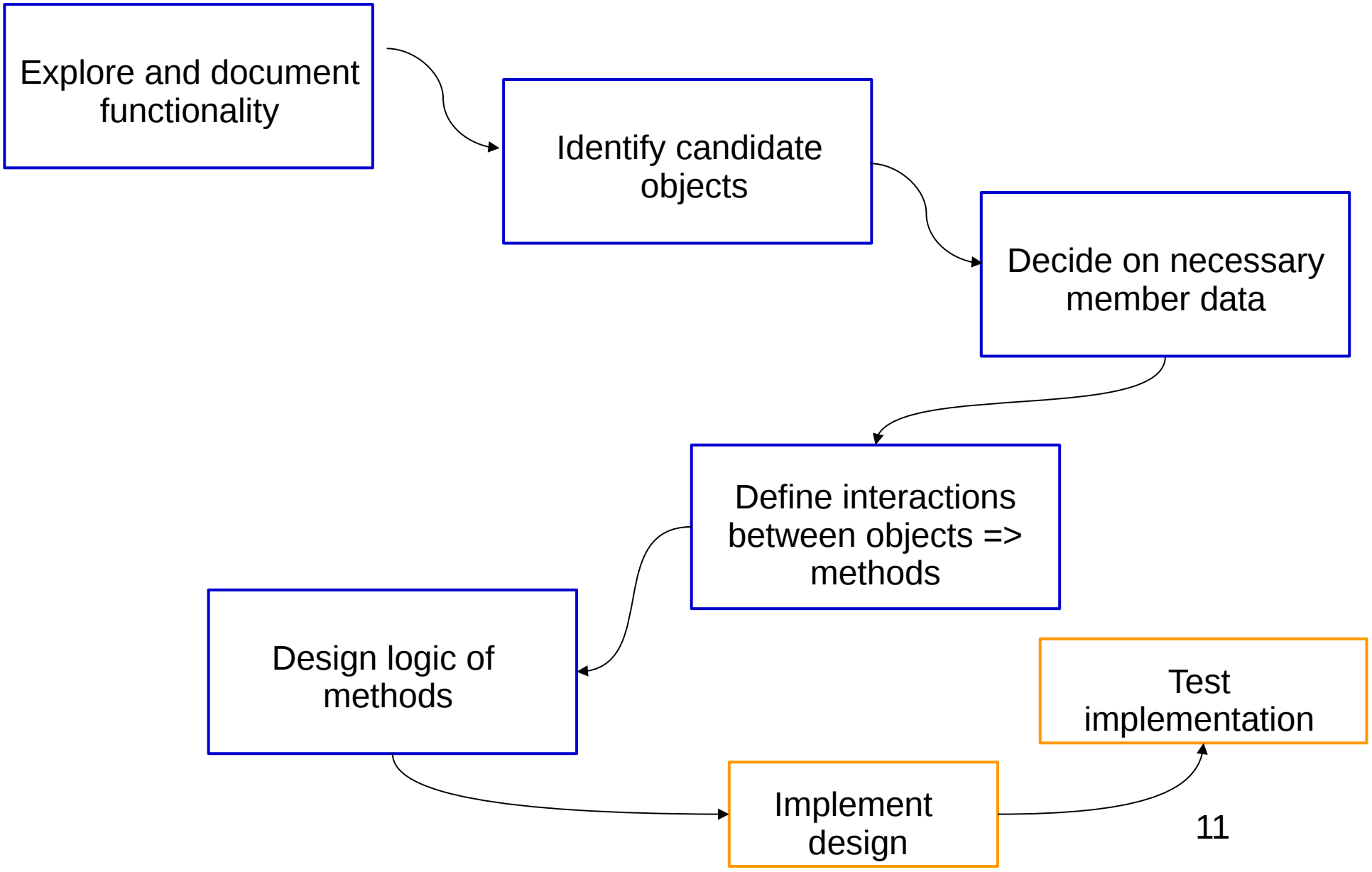
Objects = Data + Behavior

EmailMessage object

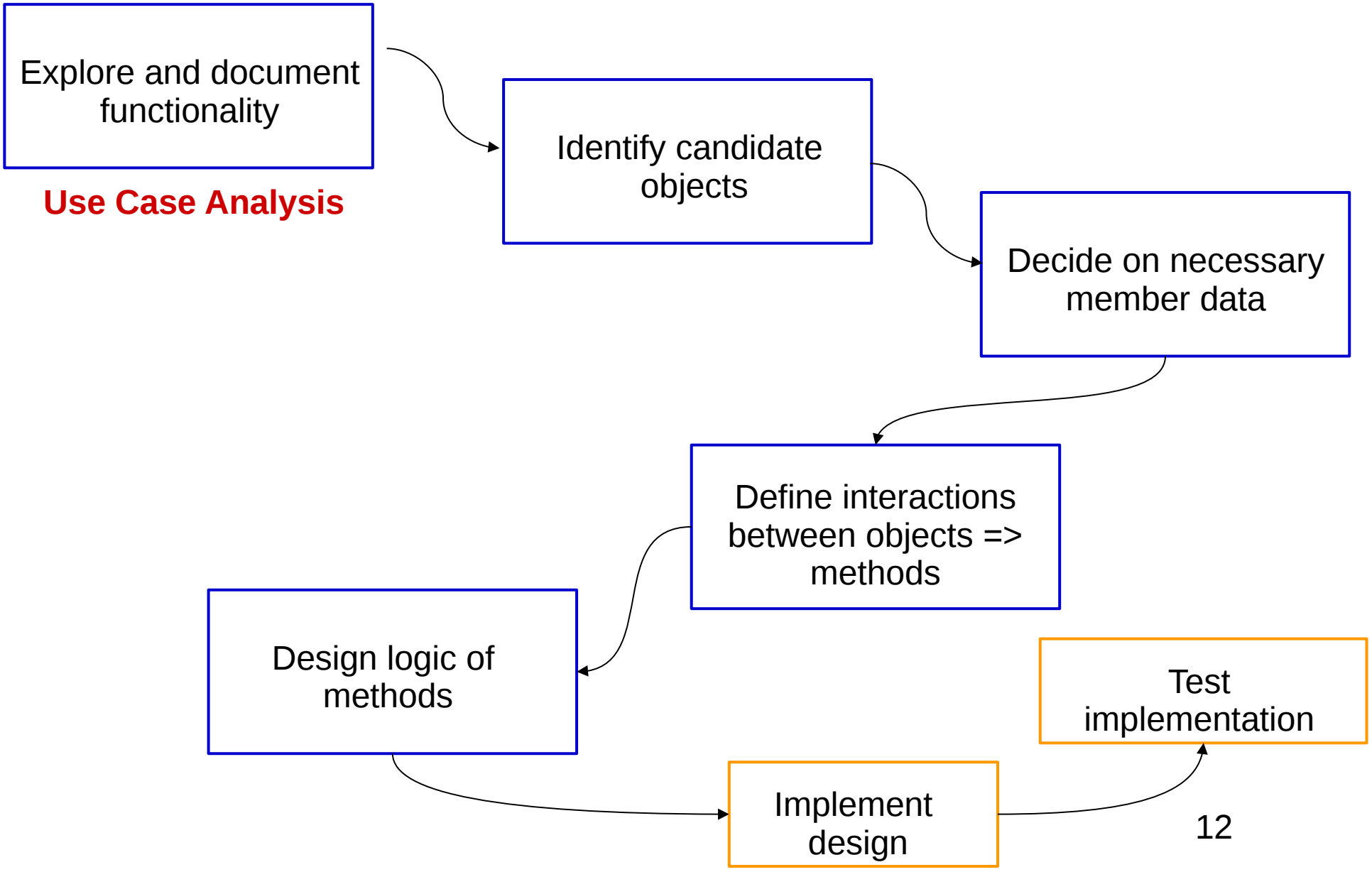
- Data (*members*)
  - toAddress
  - fromAddress
  - timestamp
  - messageBody
  - attachedFiles
  - Etc.

- Behavior (*methods*)
  - compose
  - edit
  - send
  - display
  - forward
  - delete
  - move
  - Etc.

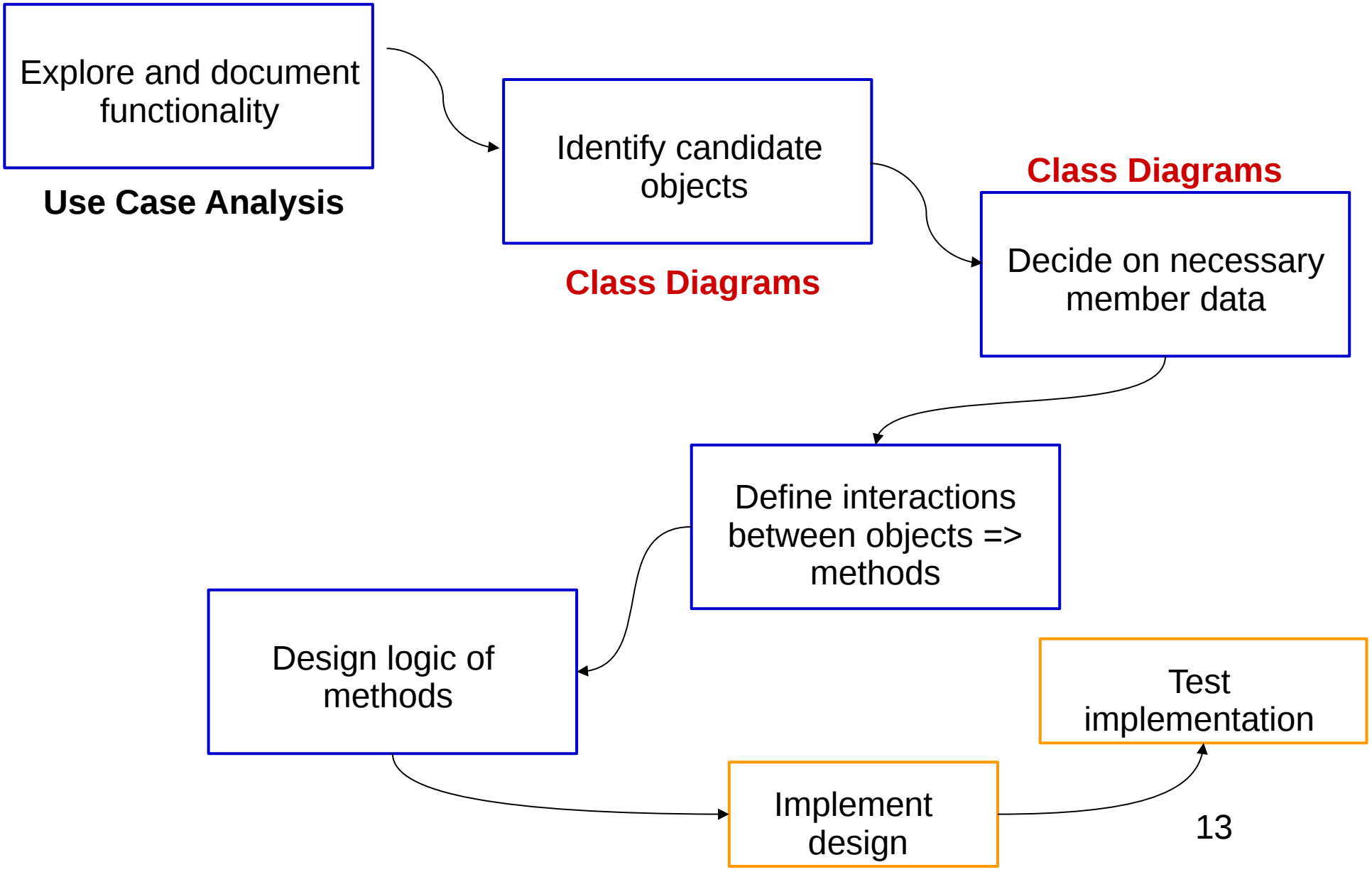
The OOD Process



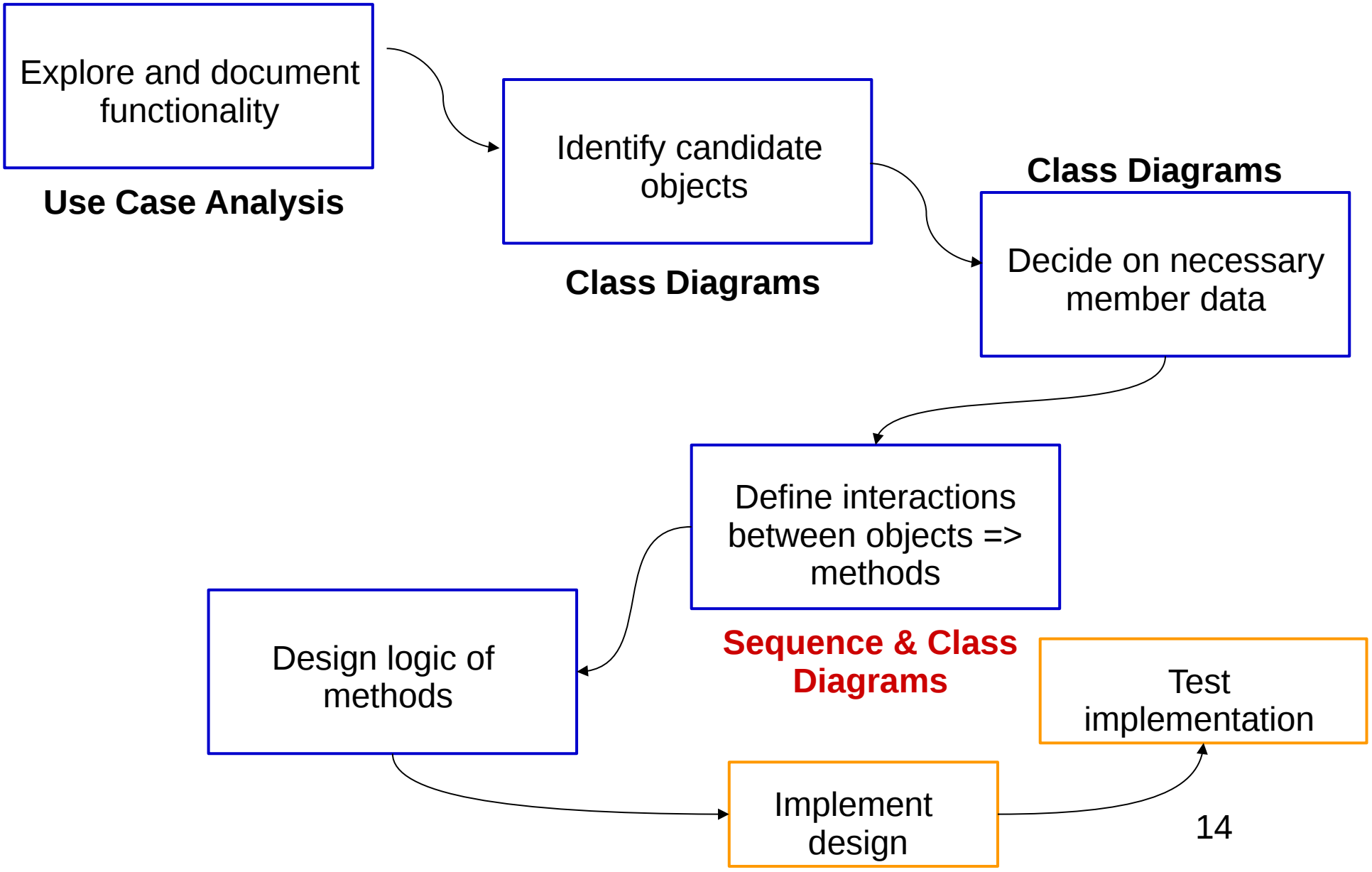
The OOD Process



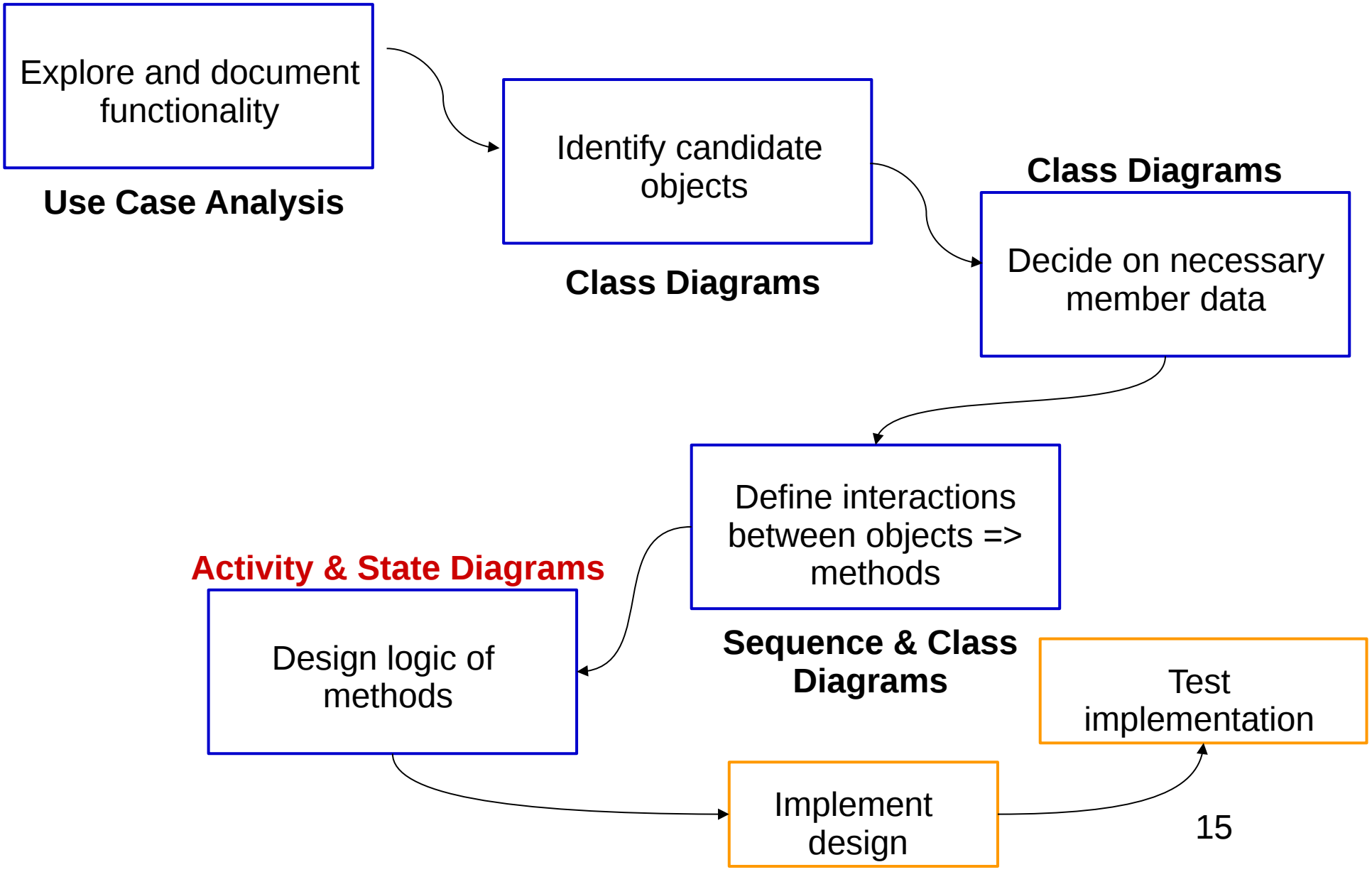
The OOD Process



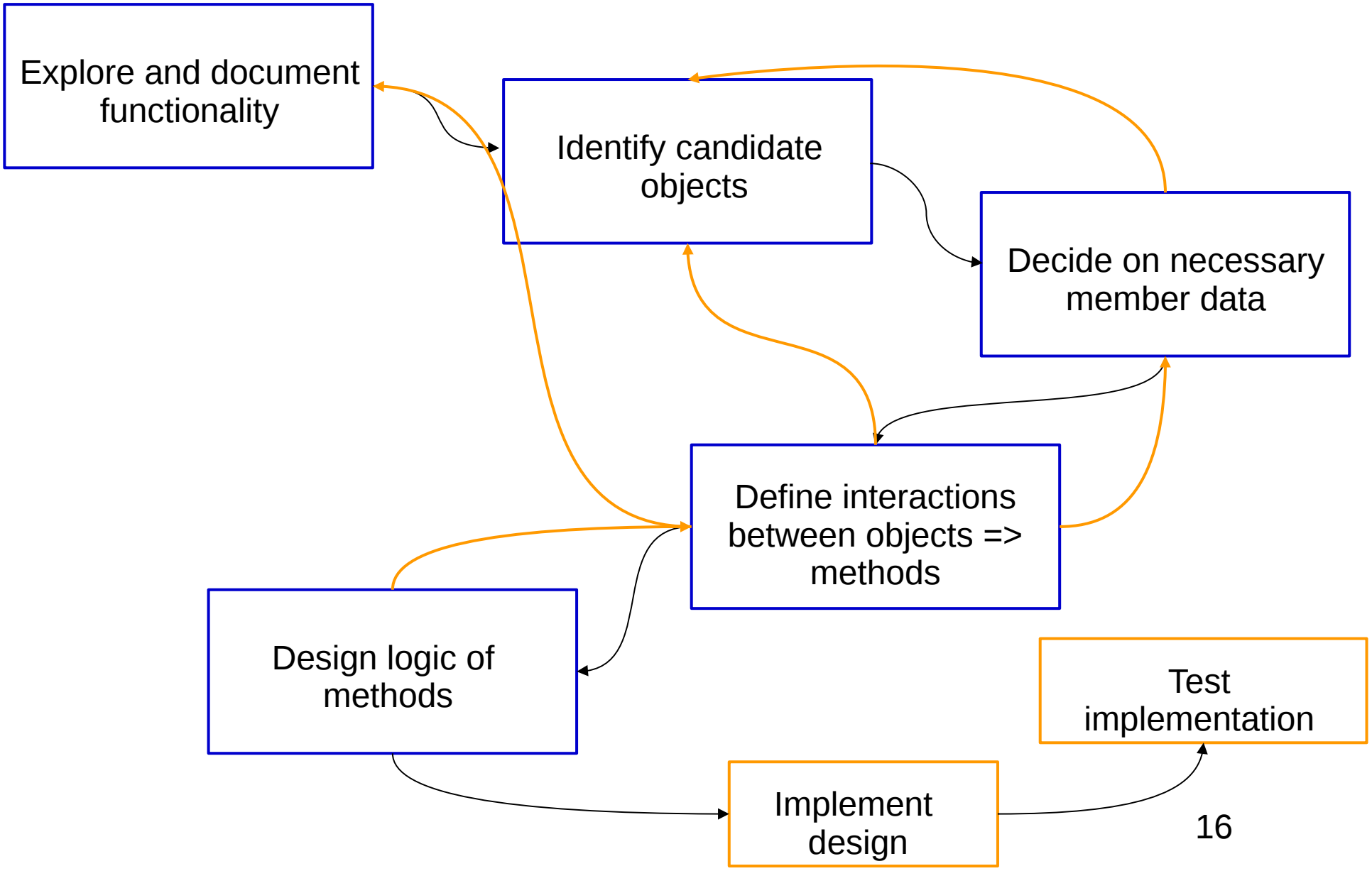
The OOD Process



The OOD Process



OOD tends to be iterative



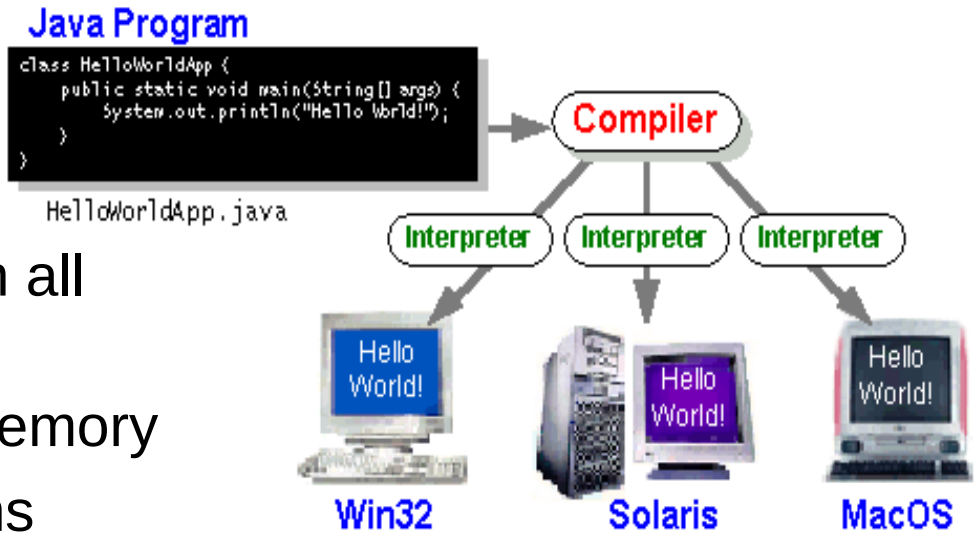
Introducing Java

- Java is a relatively new (compared to C!), object oriented language that has become very popular since it was first introduced in the late 1990s.
- Java was invented by James Gosling at Sun Microsystems, a company that was purchased by Oracle in the mid-2000's. Java is freely available but it is not open source.

  (See https://www.oracle.com/java/technologies/)
- Java is currently at Version 13. However, for compatibility purposes we will use Java 8 in this course. You should download and install it on your computer, from here:

https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

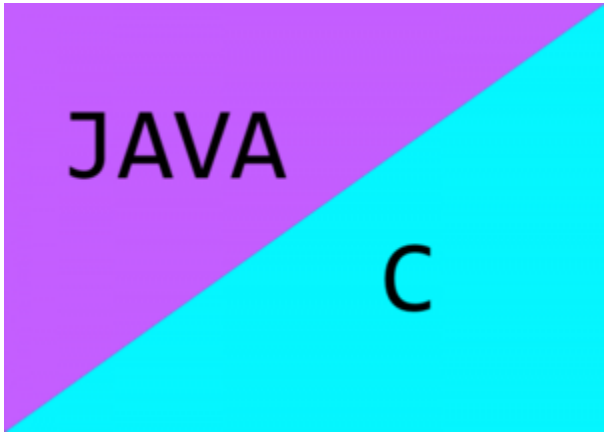
Important Java Features



- “Write once, run anywhere"
- Imposes a consistent structure on all programs
- Built-in capability for managing memory
- Built-in error handling mechanisms
- Rich set of system libraries for building modern software applications
- Can be used for many different types of applications
    - standalone desktop applications
    - client server applications
    - mobile applications
    - web applications
    - embedded applications.

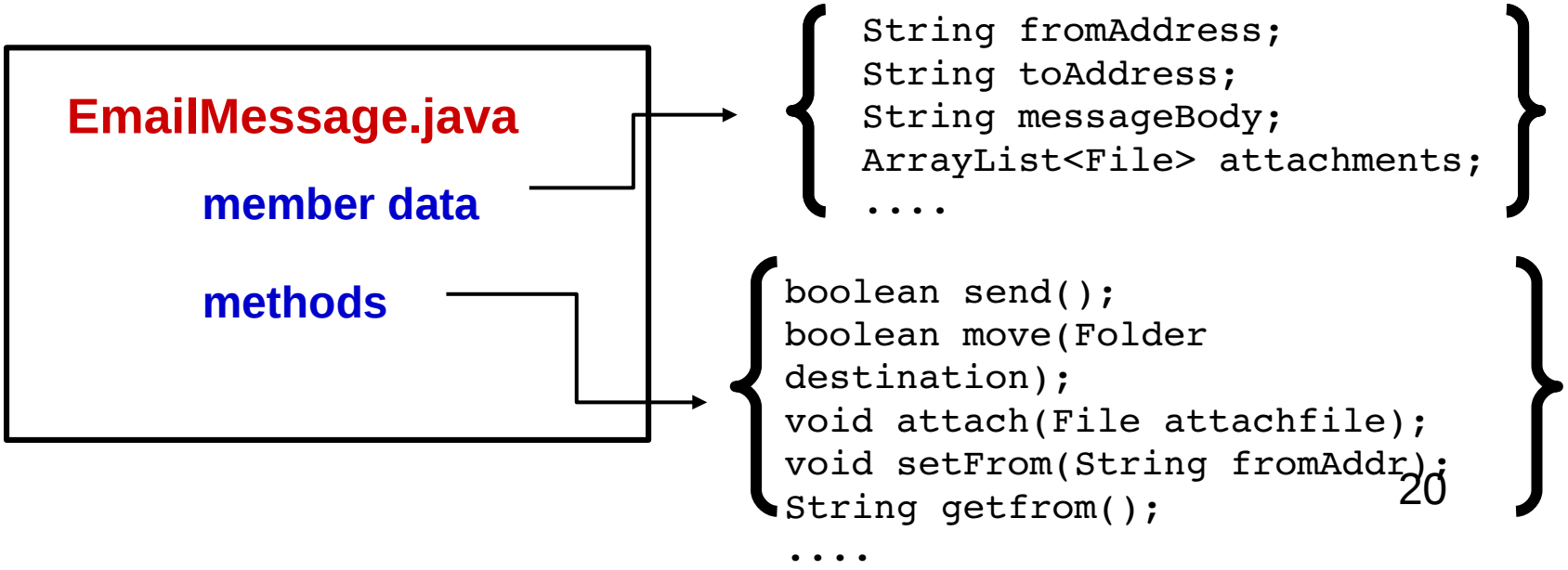
Java has a lot in common with C

- Identifiers, variables, operators & expressions
- Similar primitive types (`int, double, char`) – also `boolean`
- Arrays (though syntax slightly different)
- Flow of control (conditionals, loops, switches)
- Other keywords: `break`, `continue`, `return` etc.
- Blocks delimited by {}, similar variable scope rules



Every Java source file defines a “class”

- A class defines a category or type of objects
- That definition includes:
  - Data items specific to that type of object (*member data*)
  - Functions that operate on that data or perform actions necessary for the class (*methods*)



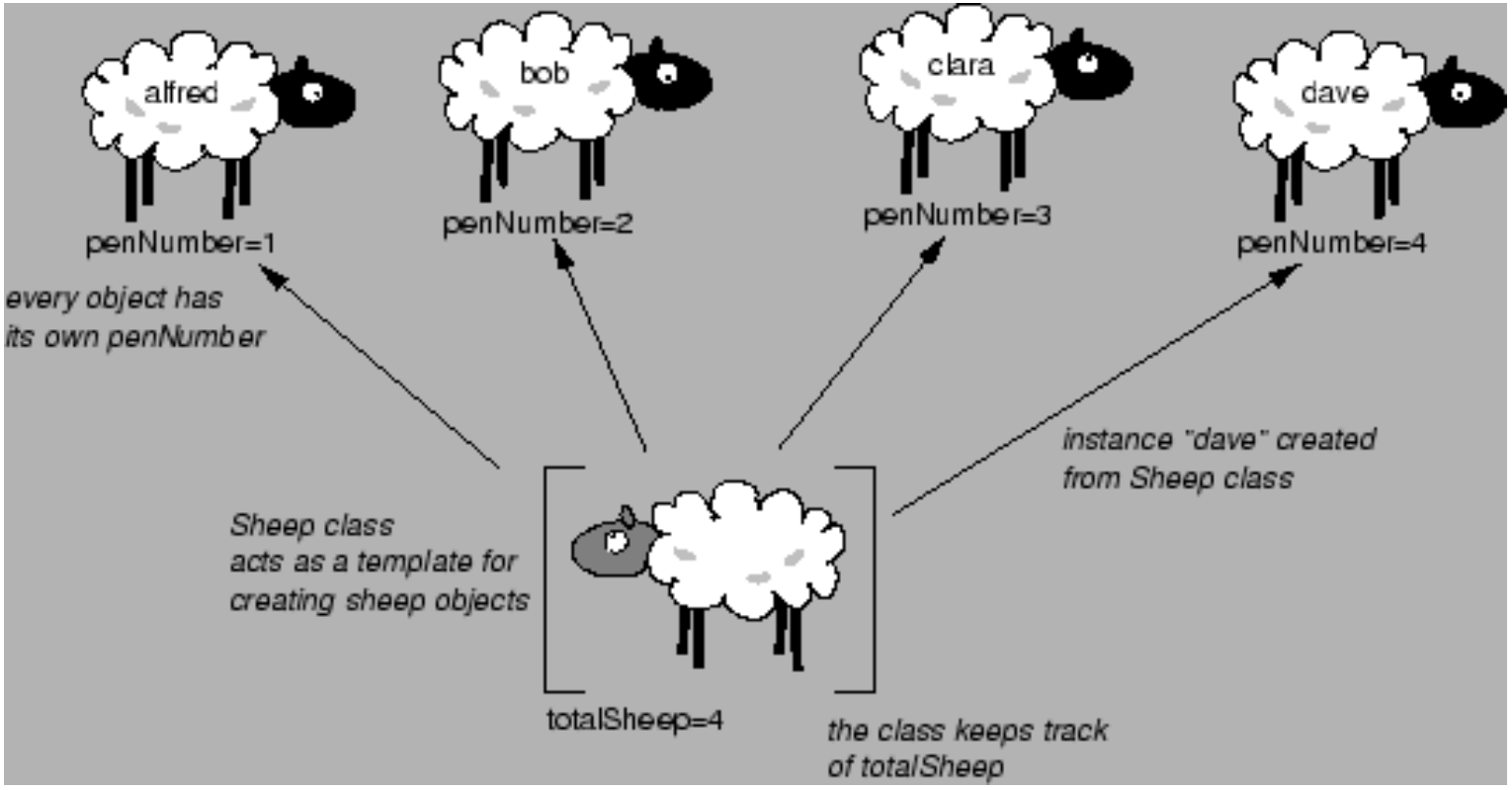
Constructors

- Most Java classes include a special method called a *constructor*
- The constructor is used to create *instances* of the class, that is, specific objects that belong to the object category
- The constructor is a function with the same name as the class and no explicit return value. It may or may not have arguments.
- To create a class instance, you call the constructor with the keyword `new`. Normally you assign the result to a variable with the type of the class.
- Then you can use that variable to call other methods.

```
EmailMessage myMessage = new EmailMessage();
....
myMessage.setFrom("seg@goldin-rudahl.com");
```

Classes and Instances

Every instance of a class has the same member variables.
However, the values stored in each of these variables will likely be different.



Sheep.java has member variables sheepName and penNumber

Note: most object oriented languages have this distinction between classes and instances. This is not specific to Java.

Structure of a Java Program

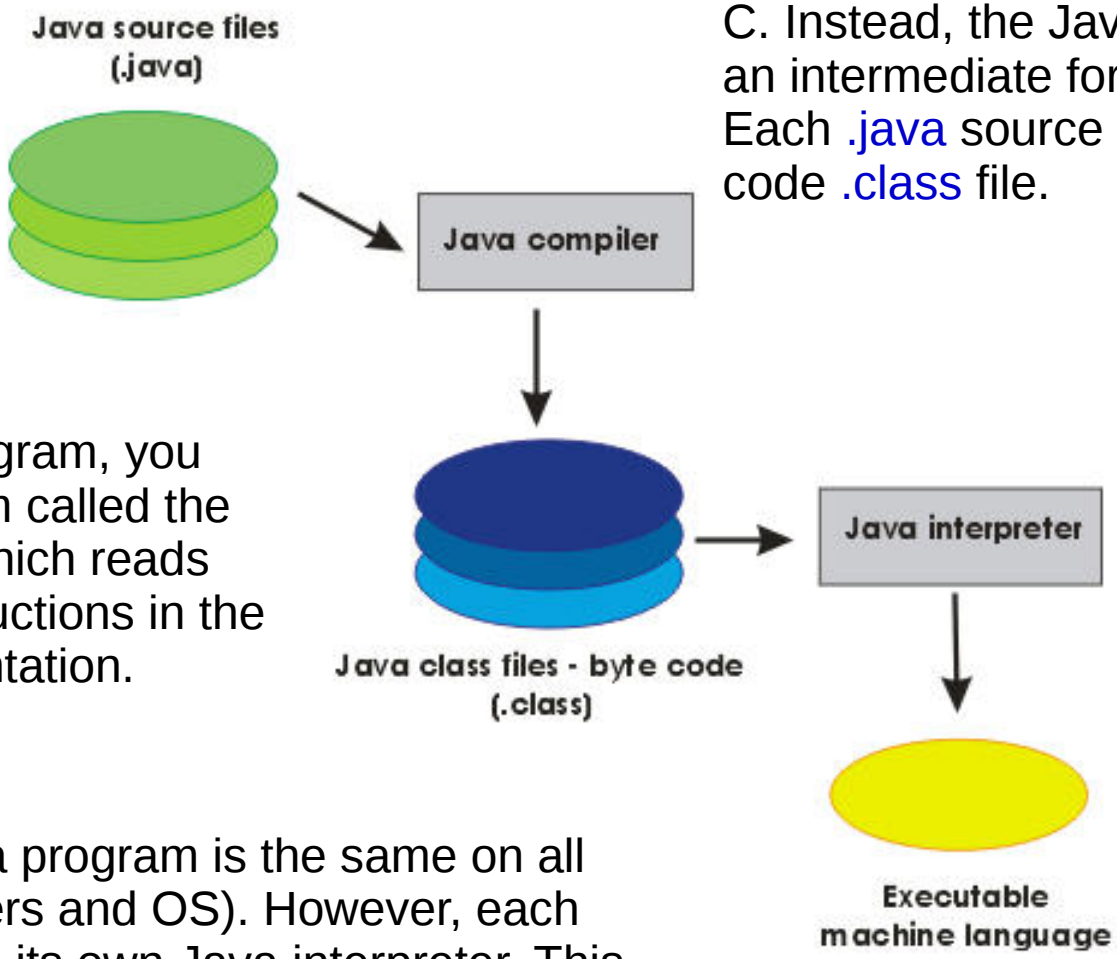
- A Java program is a collection of classes
- Each class must be defined in a separate source file
- The source file must use the naming convention *ClassName.java* (for example, *EmailMessage.java*)
- At least one class must have a special method called **`main()`**.

```
public static void main(String args[])
```

- As in C, execution of the program will start at that method.
- The **`main()`** method will create instances of classes, call methods, and so on.
- Often this method will be located in the “top-level” class of the program. If your program has a graphical user interface (GUI), this will usually be a class that represents the main window of your GUI.

Building and Running a Java Program

Java source files are not compiled directly into an executable form like C. Instead, the Java compiler creates an intermediate form called *byte code* Each .java source file creates a byte code .class file.



To execute the program, you must run a program called the *Java interpreter*, which reads and executes instructions in the byte code representation.

The byte code for a program is the same on all platforms (computers and OS). However, each platform must have its own Java interpreter. This is how Java can be "Write once, run anywhere".

Command Line Tools

- The command to compile a Java source file is **javac**

```
javac EmailMessage.java
```

- You must include the file suffix. You can compile multiple source files at once:

```
javac *.java
```

- The command to run the Java interpreter is **java**

```
java EmailClient
```

- The example above assumes that the file *EmailClient.class* includes a static *main()* function. You must **not** include the *.class* suffix!
- Java applications tend to have lots of source files. Thus it is common to use either a Makefile or an Integrated Development Environment (IDE) like Eclipse or NetBeans when doing Java programming.

In this course...

- You will be writing a lot of Java code
- You will mostly need to learn Java syntax and details on your own (though you can always ask questions)
- Resources:
  - Slides from Aj. Natasha's Java class are in the Java Background directory of the class home page. (Note these slides are quite old, so there are some things that have changed, but they cover a lot of material.)
  - Java has excellent online documentation. The API docs for Version 8 are here: http://docs.oracle.com/javase/8/docs/api/
  - The Oracle website also has lots of tutorials and demos
  - My favorite Java reference book is *Java in a Nutshell* which is available from O'Reilly Publishing:

http://shop.oreilly.com/product/0636920030775.do

Java Coding Standards

- The Java code you write in this course must conform to my coding standards or you will lose significant credit
- These standards are similar to the C coding standards, with a few changes and additions
- You can find a link to these rules on the course home page.

Assignments

1. Do Exercise 1. We'll start working on this in class. It is due by Monday at 17:00. Use the upload link on the home page to submit your work.
2. Join the course Facebook group.
3. Create teams for the term project. Each team should have **two** people. Next Tuesday you must hand in a sheet of A4 paper with the following information:
   - *Team name*
   - *Name, student ID and nicknames of team members*