PeopleCert
Software Development Skills

JavaScript Stream

Lesson 7

Study Guide

Copyright Details

e-mail: info@peoplecert.org, www.peoplecert.org

- Document Version: PC-SDS_SG 1.0 | February 2020

PeopleCert
All talents, certified.

PeopleCert: A Global Leader in Certification

- ✓ **Web** & **Paper based** exams in **25** languages
- ✓ **2,500 Accredited Training Organisations** worldwide
- ✓ Delivering exams across **200 countries every year**
- ✓ Comprehensive Portfolio of **500+ Exams** and Growing

AXELOS GLOBAL BEST PRACTICE · PeopleCert DevOps. · digital marketing institute · IASSC · ISACA · Language Cert · LEAN IT ASSOCIATION · Quality Software Development · ECDL

PeopleCert
All talents, certified.

How to Use This Document

This document is your **PeopleCert Software Developer Skills Study Guide** to help you prepare for the **PeopleCert Software Developer Skills Foundation & Advanced examination**.

It is meant to provide you with a clear outline of everything covered in the course presentation by your instructor that will be on the PeopleCert Software Developer Skills Foundation & Advanced exams.

Your exams will be closed book. You will be given 120 minutes to complete it. It contains 100 multiple choice questions and to pass the exam you must achieve a grade of 65% or higher, or a minimum of 65/100 correct responses. For further details on your exam, including more information on question types and learning objectives, please refer to your course syllabus.

As you follow along, you may see that some material here is not replicated in the trainer presentation. This study guide includes questions, activities, knowledge checks, or other material in the presentation that are facilitated verbally by the instructor. It also does not contain content that is not examinable, but instead is designed to reinforce learning or add value to your course experience. It also provides valuable links and references, throughout the slides, which you can explore further to enhance your learning and understanding of the material provided in the study guide.

PeopleCert
All talents, certified.

Coding Bootcamp

LESSON 7

Software Design and Development

Objectives

- Software development process & activities
- Software design process
- Requirements Capturing
- UML Diagrams

Syllabus Items:

- 1.4 Requirements Capturing and Software Design

Syllabus

Category	Topic	Task	
FSD_1 Software Design and Development	**1.4 Requirements Capturing and Software Design**	1.4.1	Understand the process of requirements capturing and know how to complete it
		1.4.2	Understand the process from requirements capturing to software design and apply it to a simple case
		1.4.3	Create a mock-up design
		1.4.4	Use Unified Modelling Language (UML) to create a Use-case diagram
		1.4.5	Use Unified Modelling Language (UML) to create a Class diagram
		1.4.6	Use Unified Modelling Language (UML) to create a Sequence diagram, State transition diagram

Contents | Learning Objectives

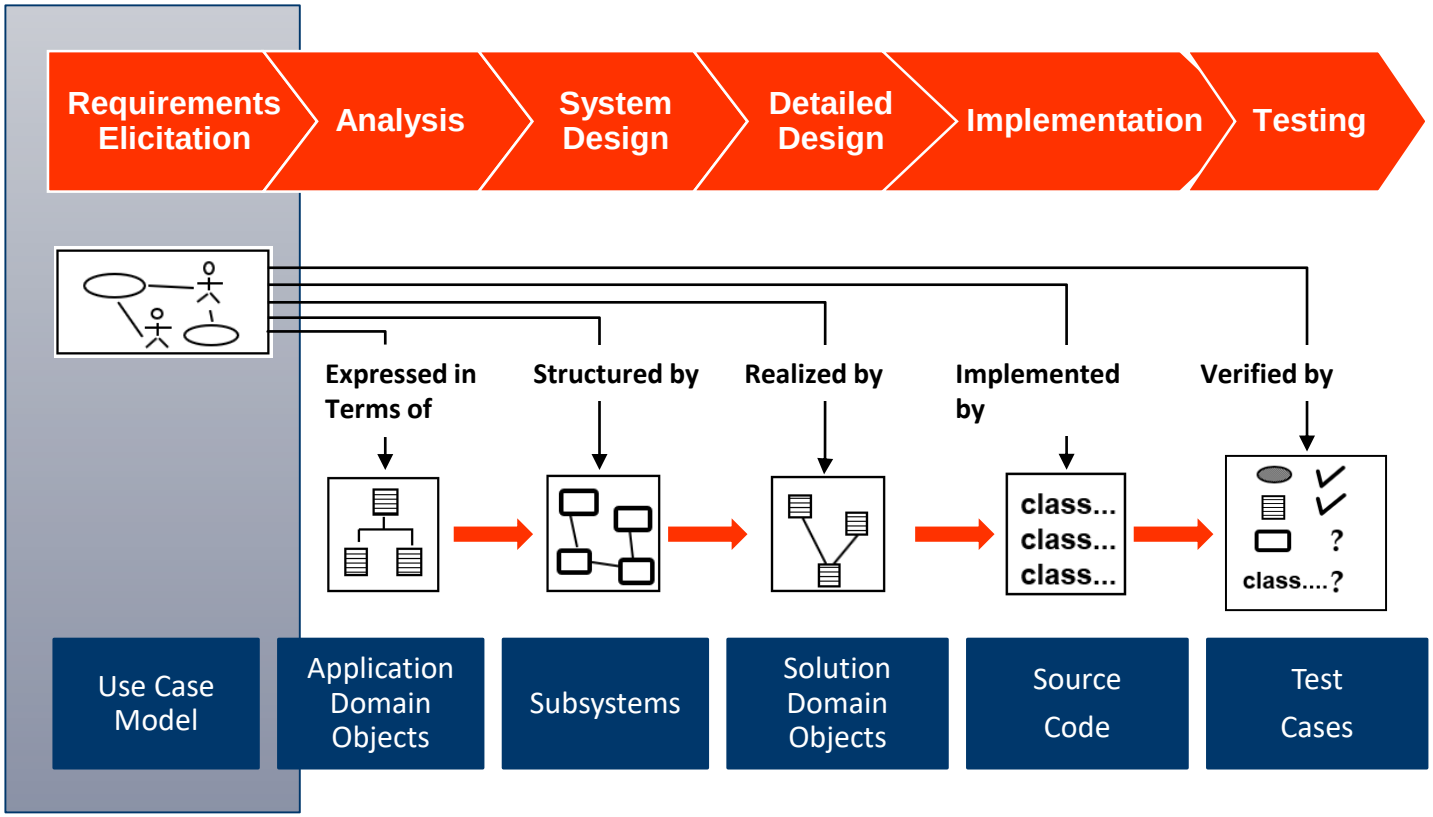
- ✓ Understand the software development lifecycle activities
- ✓ Understand the software design process
- ✓ Know what is requirements capturing and how to complete it
- ✓ Understand software design is derived from requirements capturing to software design and know how to apply (simple case)
- ✓ Use Unified Modelling Language (UML) to create various diagrams used in software development, like a Use-case diagram, a Class diagram, a Sequence diagram, State transition diagram

Slide Title

- Regular Content

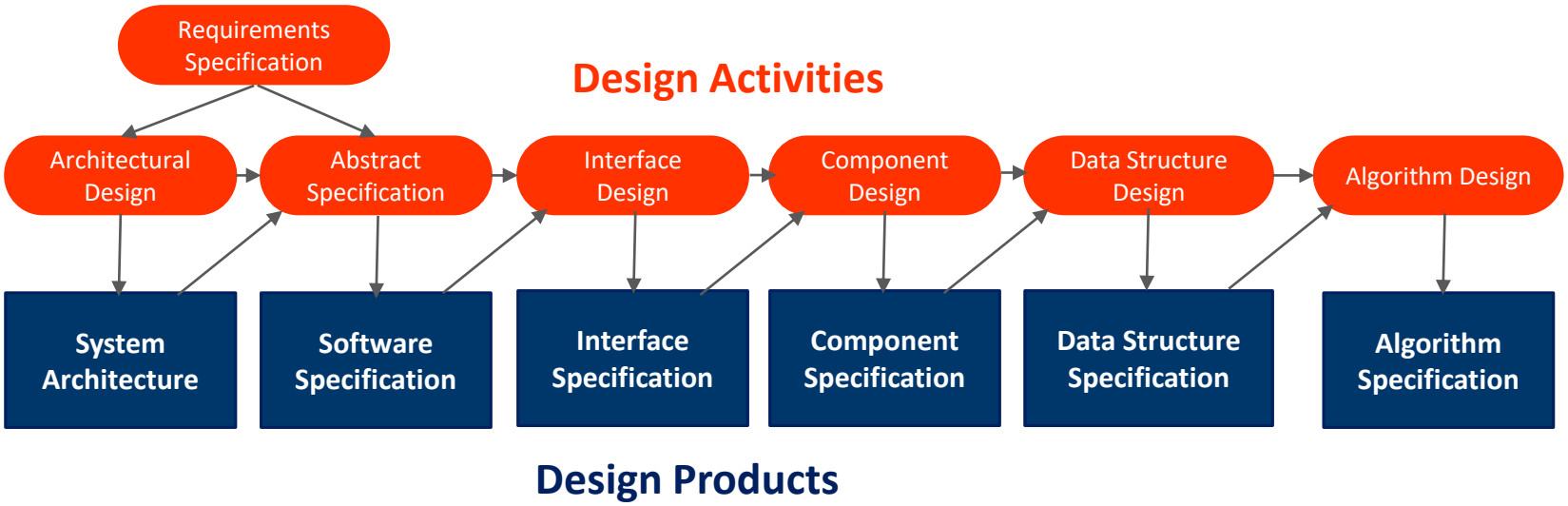
Source: XXXXX

Software Lifecycle Activities



Adapted from: https://iansommerville.com/software-engineering-book/

The Software Design Process



Requirements Capturing

- *Determine the needs or conditions to meet taking account of the possibly conflicting requirements of the various stakeholders, analyzing, documenting, validating and managing software or system requirements*

User Requirements

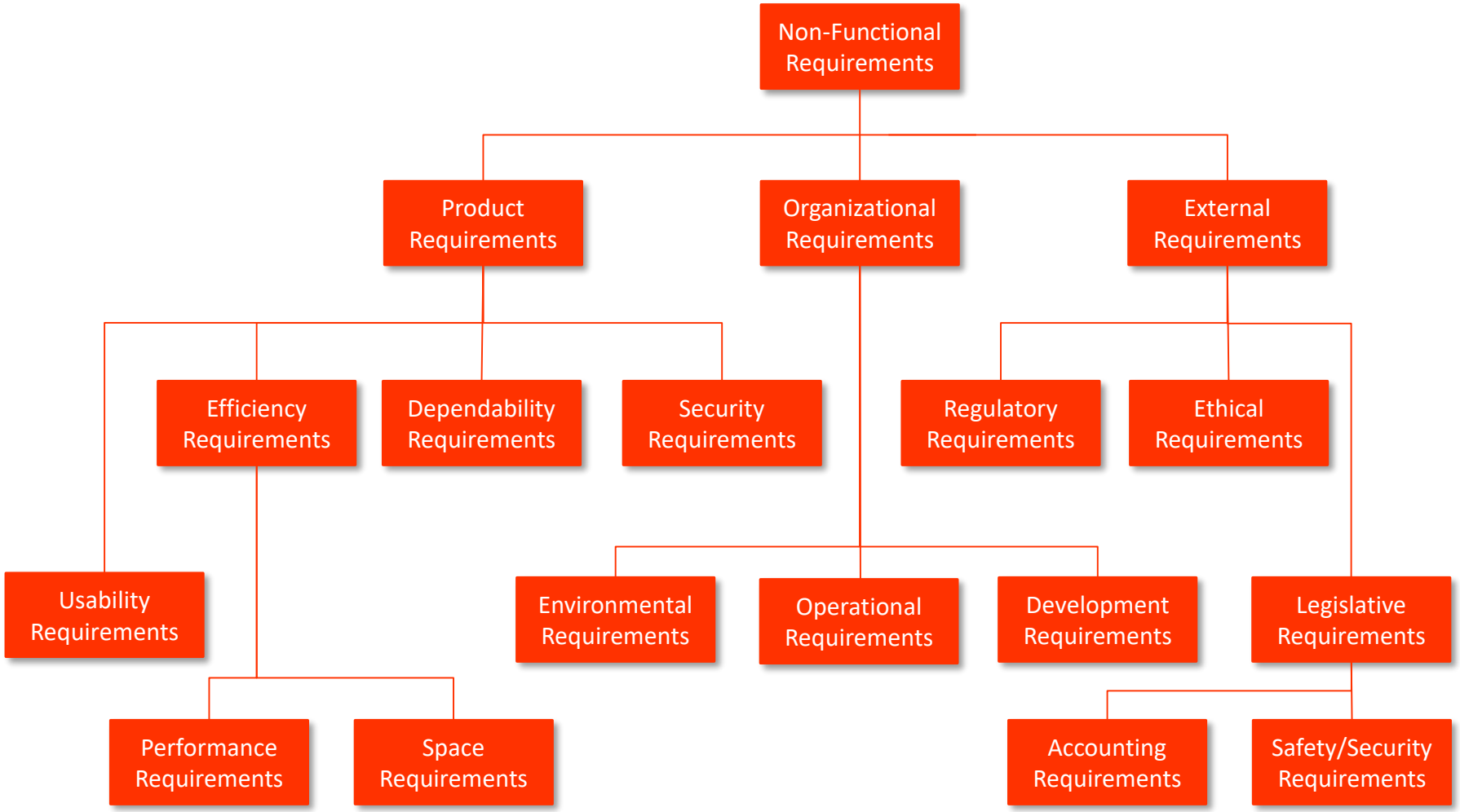
- Client Managers
- System End-Users
- Client Engineers
- Contractor Managers
- System Architects

System Requirements

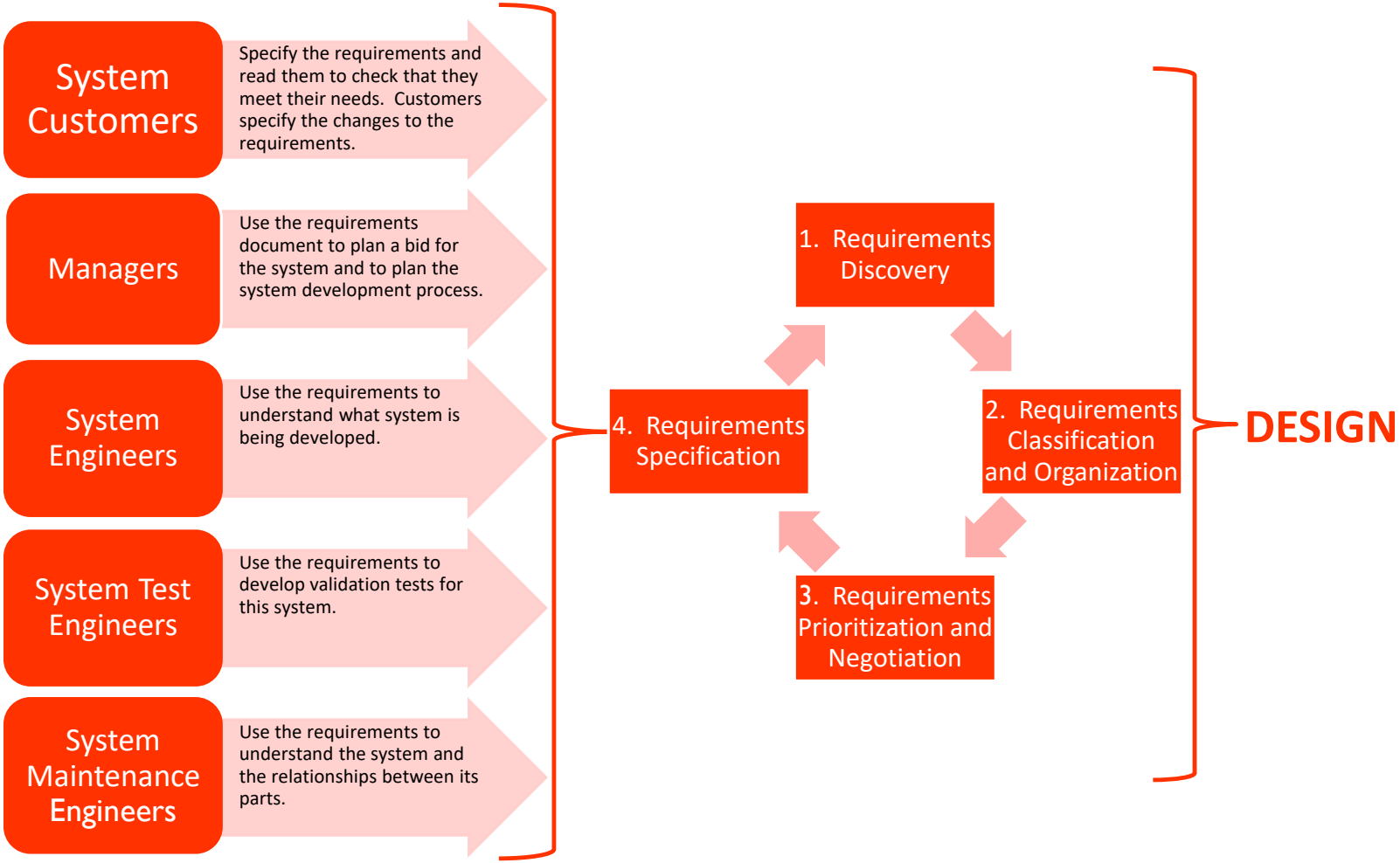
- System End-Users
- Client Engineers
- System Architects
- Software Developers

Sources: https://iansommerville.com/software-engineering-book/
https://en.wikipedia.org/wiki/Requirements_analysis

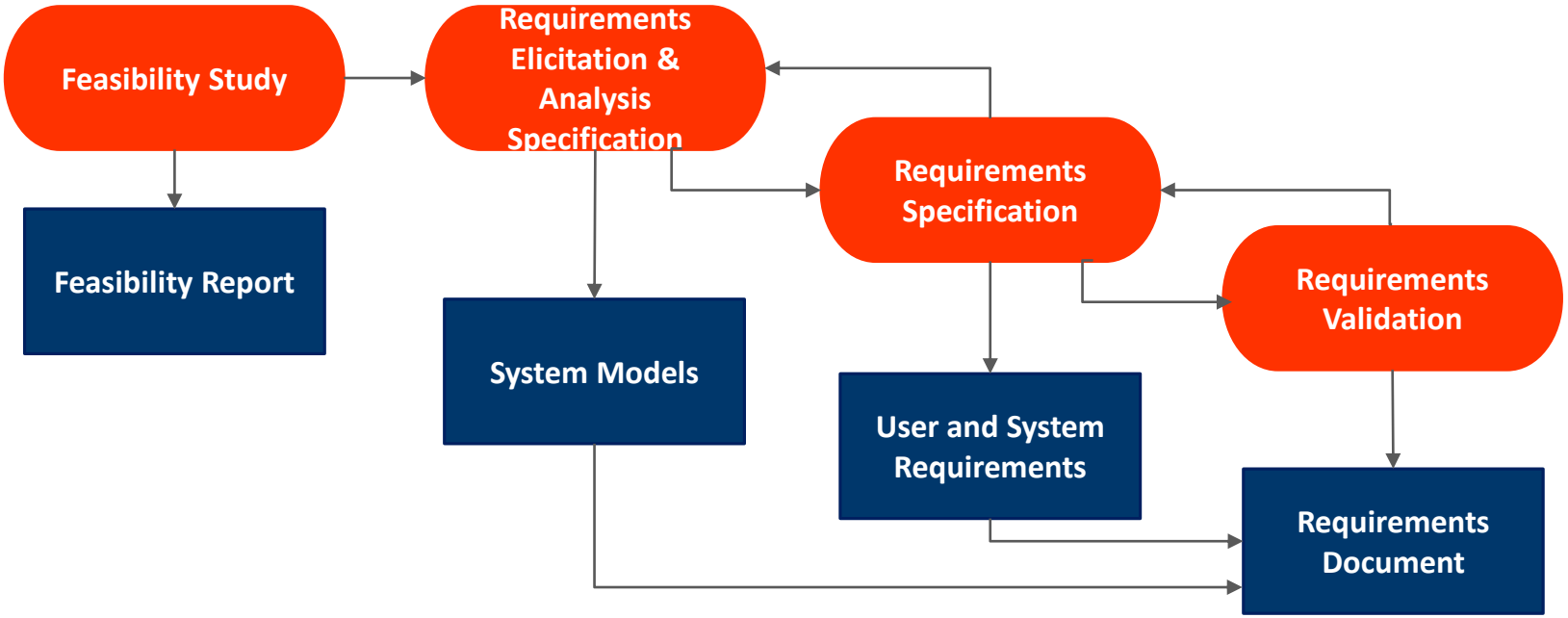
Requirements Capturing



From Requirements Capturing to Software Design



The Requirements Engineering Process



5 Principles to Good Requirements

1. **Communicate Input to Design**
   - What are we solving?
   - Why is this function important?
   - Clarity to Cross-Functional Team
2. **Measurable & Testable**
   - Verification and Validation are Possible
   - Subjective Requirements cannot be Verified
3. **Requirements are Focused**
   - Audience for Requirement is known
4. **Provide Value to Development**
   - Based on Need: Answer WHY?
5. **Free of Specific Design Content**

Class Exercise 1

Consider a simple game, like tic-tac-toe

- Draw a flowchart showing how the game works
- What are some basic requirements?

Unified Modelling Language (UML)

- Use Case Diagram
- Class Diagram
- Sequence Diagram
- State Transition Diagram

Use Case Diagram

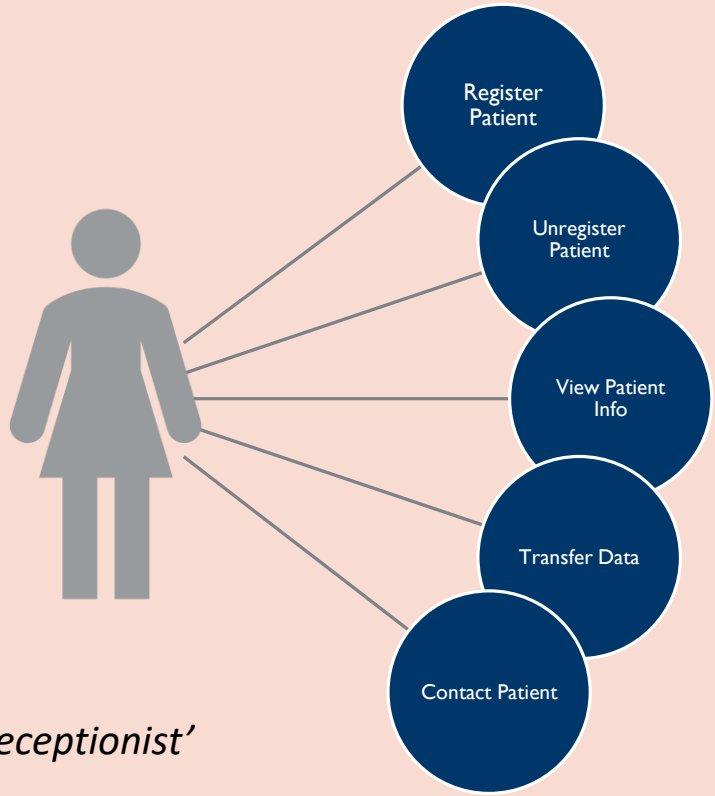
- A **use case diagram** is "a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved".

Transfer-Data Use Case Diagram Example

Source: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/
Adapted from: https://iansommerville.com/software-engineering-book/

Class Exercise 2

- Using the previous diagram as an example, draw a Use Case Diagram showing:
- A medical receptionist
- 4 Use Cases (any from the below list)
  - Register patient
  - Unregister Patient
  - View Patient Info
  - Transfer Data
  - Contact Patient

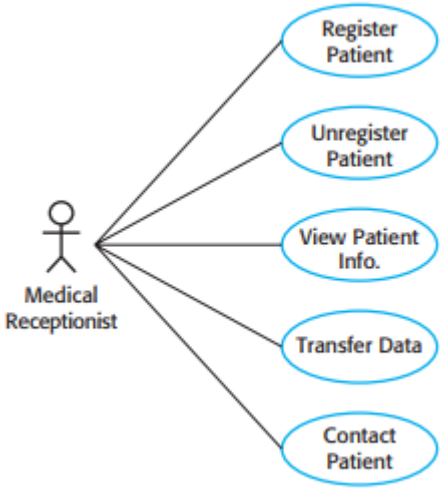


Use cases involving the role of 'medical receptionist'

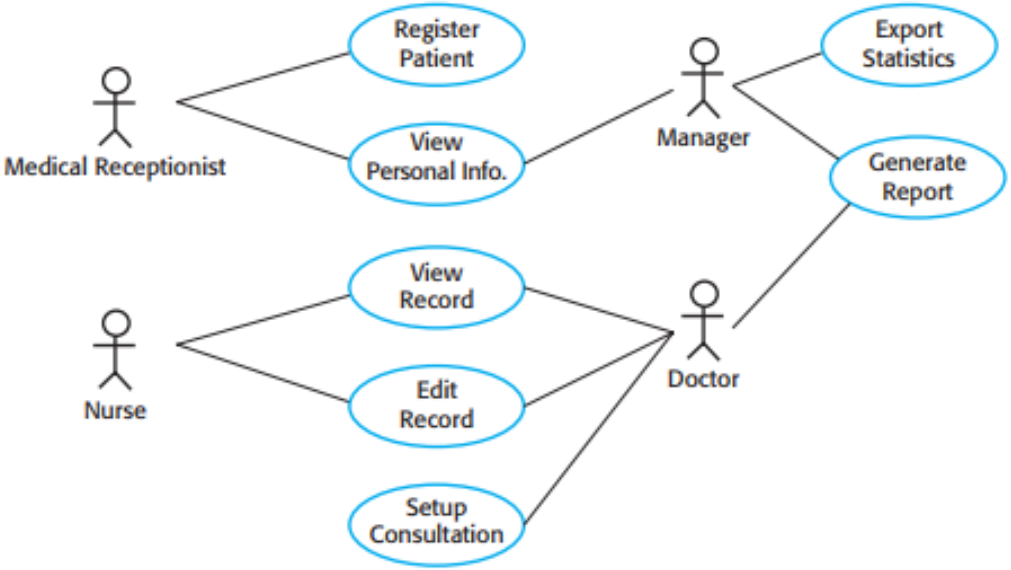
Adapted from: https://iansommerville.com/software-engineering-book/

Use Case Diagram (Discussion)

- **Discussion**

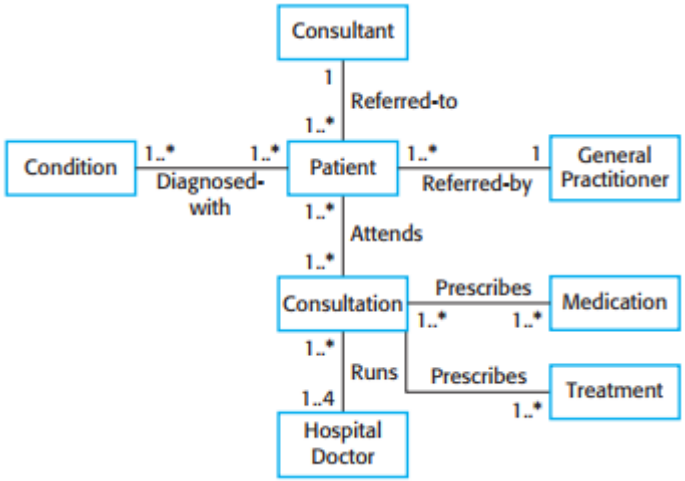


- **Solution**



Source: XXXXX

Class Diagram

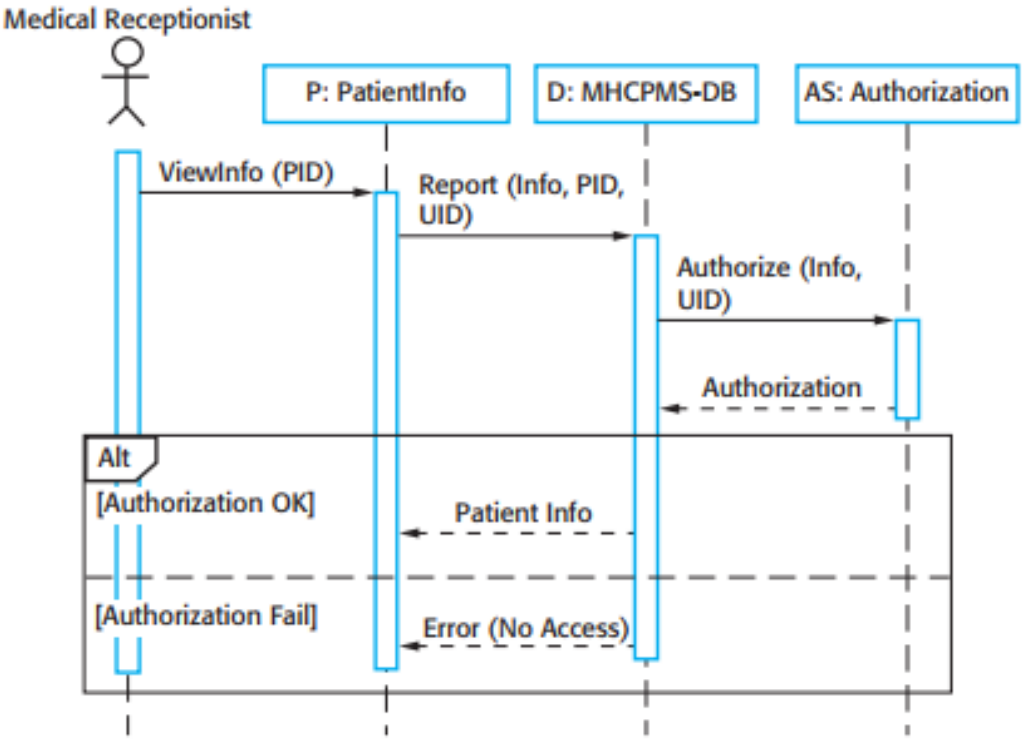


Classes and Associations Example

Source: https://iansommerville.com/software-engineering-book/

Sequence Diagram

- **A sequence diagram** is an interaction diagram that shows how objects operate with one another and in what order



Source: https://iansommerville.com/software-engineering-book/

State Transition Diagram

State Diagram of a microwave oven (Example)

Source: https://iansommerville.com/software-engineering-book/

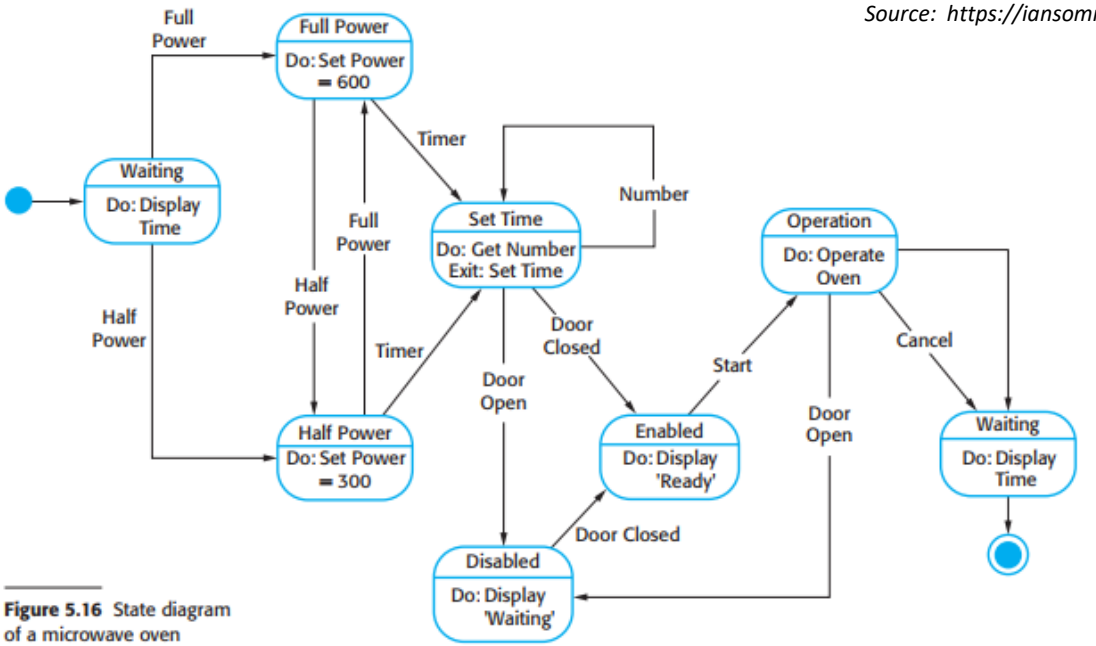


Figure 5.16 State diagram of a microwave oven

- A **state diagram** is a type of diagram used in computer science and related fields to describe the behavior of systems.

Source: https://en.wikipedia.org/wiki/State_diagram

Software Development Tools

- Application Development Frameworks
- Content Management Systems (CMS)
- Open source e-Commerce platforms
- Software development tools

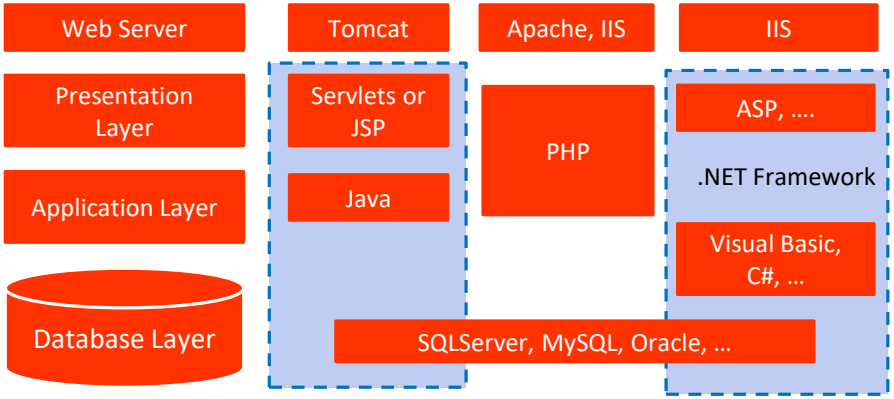
Application Development Frameworks

Definition:

*An **application framework** consists of a software framework used by software developers to implement the standard structure of application software.*

Application frameworks became popular with the rise of graphical user interfaces (GUIs), since these tended to promote a standard structure for applications. Programmers find it much simpler to create automatic GUI creation tools when using a standard framework, since this defines the underlying code structure of the application in advance.

Developers usually use object-oriented programming (OOP) techniques to implement frameworks such that the unique parts of an application can simply inherit from classes extant in the framework



CMS

Definition:

A content management system, or CMS, is a web application designed to make it easy for non-technical users to add, edit and manage a website.

Its basic functions are:

- Automatically generate navigation elements
- Making content searchable and indexable
- Keeping track of users, their permissions and security settings
- and much, much more

Examples:

Source: https://en.wikipedia.org/wiki/Application_framework

Open Source e-Commerce Platforms

Definition:

*An **e-commerce platform** is a software application which allows the business owner to manage their online sale operations.*

There are many vendors in the market who offer different ecommerce platform which suites their business. Every platform has its own advantage and disadvantages, people chose one according to their requirement.

Examples: Shopify, Magento, Open Cart, Zen Cart, WooCommerce, Drupal, etc.

Source: http://www.webbazaar.com/ecommerce-development/what-is-an-ecommerce-platform.html

Software Development Tools

Definition:

A computer program that software developers use to create, debug, maintain, or otherwise support other programs and applications.

Examples:

- IDE (Integrated development environment)
- Netbeans
- Eclipse
- .NET
- Version Repositories
- CVS (Concurrent Versions System)
- IDEs with CVS support: Emacs , Anjuta, Dev-C++, Eclipse, NetBeans, IntelliJ IDEA, wxDev-C++, Kdevelop, Aqualogic, Xcode, PhpED
- and many, many more

Source: https://en.wikipedia.org/wiki/Programming_tool

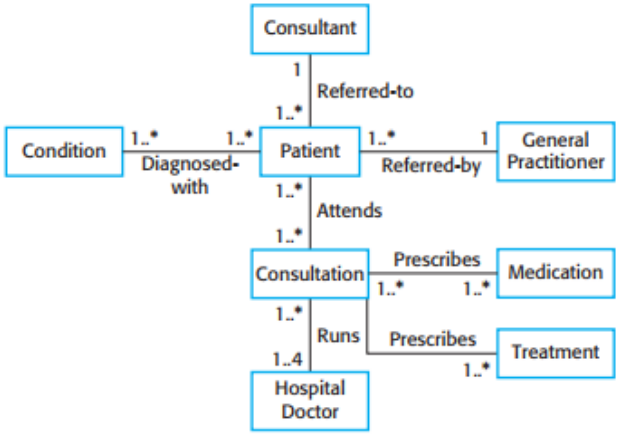
It is time for a Knowledge Check!

Sample Questions

1. "An interaction diagram that shows how objects operate with one another and in what order" is called a:
   A. Flowchart
   B. Sequence Diagram
   C. State Transition Diagram
   D. Class Diagram

2. The process of collecting the user needs to solve a problem or an issue and achieve and object is known as:
   A. Requirements analysis
   B. System analysis
   C. State Transition Diagram
   D. Class Diagram

3. The image on the right shows a(n):
   A. Sequence diagram
   B. Class Diagram
   C. Use Case Diagram
   D. State Transition Diagram

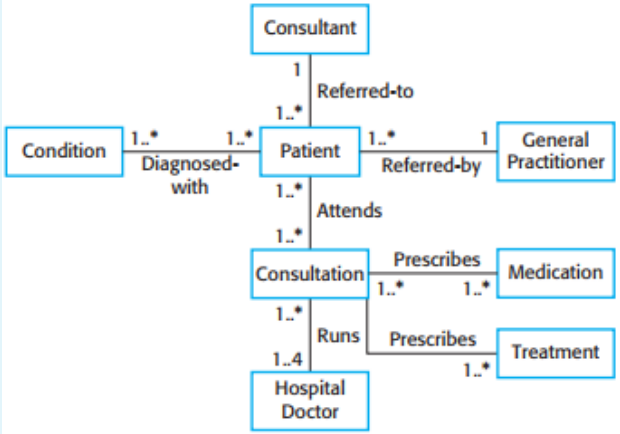


Sample Questions

1. "An interaction diagram that shows how objects operate with one another and in what order" is called a:
   A. Flowchart
   B. Sequence Diagram
   C. State Transition Diagram
   D. Class Diagram

2. The process of collecting the user needs to solve a problem or an issue and achieve and object is known as:
   A. Requirements analysis
   B. System analysis
   C. State Transition Diagram
   D. Class Diagram

3. The image on the right shows a(n):
   A. Sequence diagram
   B. Class Diagram
   C. Use Case Diagram
   D. State Transition Diagram



Coding Bootcamp

Introduction to Programming

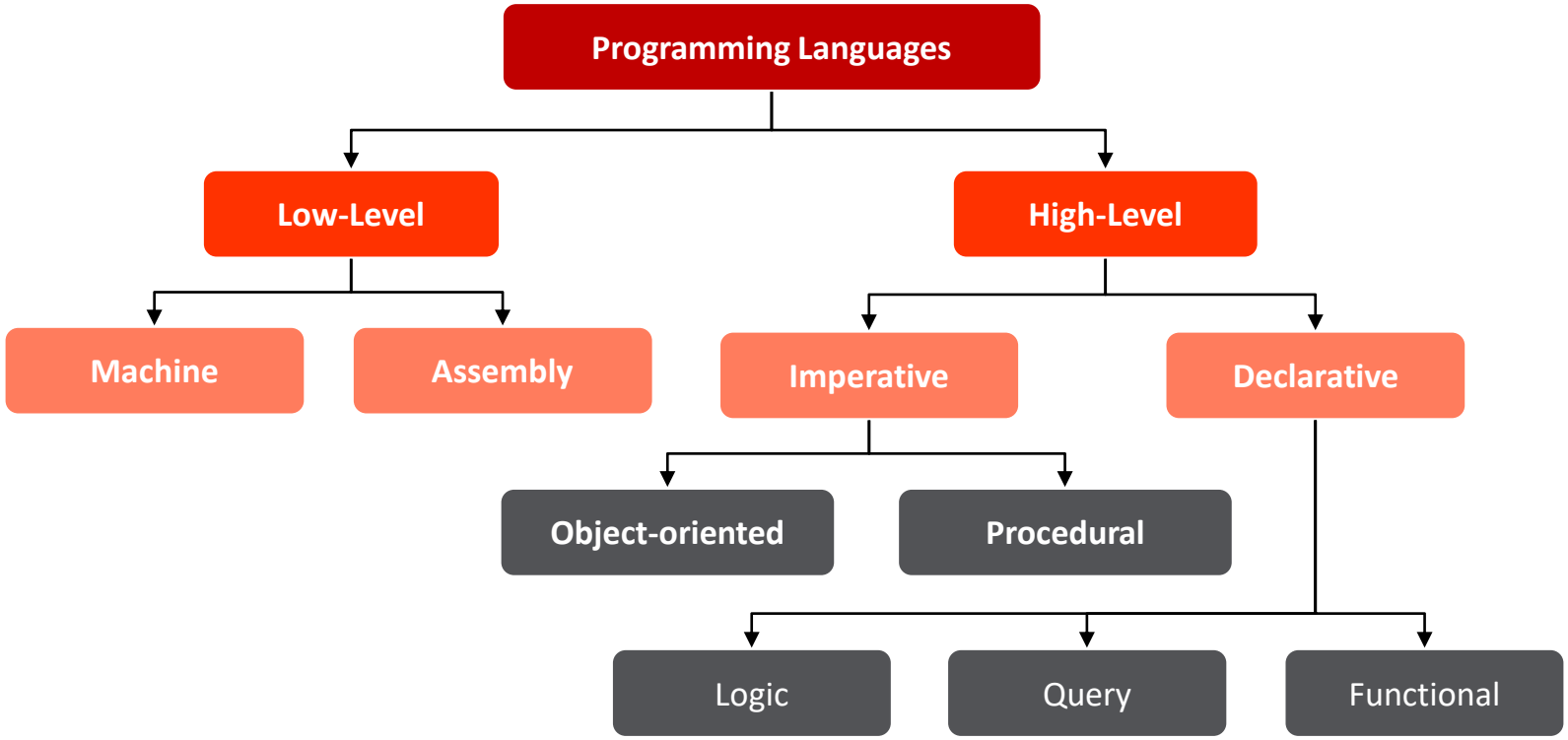
Objectives	Syllabus Topics
▪ Programming families ▪ UNIX ▪ Build Tools & IDE ▪ GitHub	▪ 2.1 Developer Tools

Category	Topic	Task
FSD_2 Introduction to Programming	**2.1 Developer Tools**	2.1.1 Understand and use editors and compilers
		2.1.2 Perform static analysis (e.g. FindBugs)
		2.1.3 Understand and use the Unix command line and its tools
		2.1.4 Use build tools (e.g. make, Maven)
		2.1.5 Understand and use IDE
		2.1.6 Understand and use GitHub
		2.1.7 Understand and use Git Version Control

- ✓ Classify computer programming languages
- ✓ Understand how the level of abstraction distinguishes programming languages
- ✓ Know the main families of programming languages, their characteristics and name some of these languages
- ✓ Know what UNIX is and know some basic commands
- ✓ Understand the difference between compiled and interpreted languages
- ✓ Understand what an IDE is and know how to use it
- ✓ Be able to create a GitHub account and use it

Classification of Programming Languages

Thousands of programming languages exist today in the market, with various purposes to fulfill. Some programming languages provide less or no abstraction from the hardware, whereas some other languages provide a higher level abstraction. To separate programming languages on the basis of level of abstraction from hardware, they are classified into various categories; however, there are two main classifications; Low Level and High Level languages.



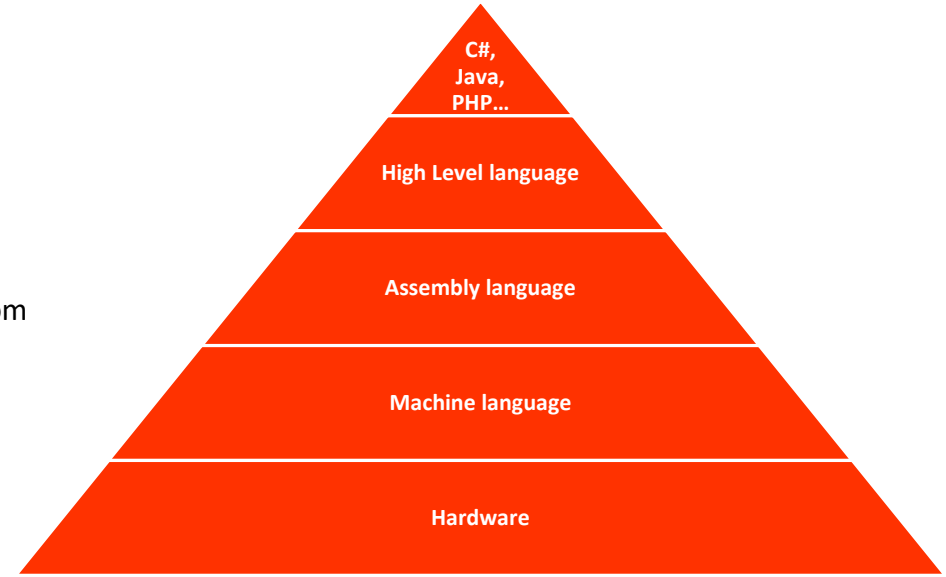
Source: https://codeforwin.org/2017/05/programming-languages-classification.html

Programming Languages

The three major families of languages are:

- Machine languages
- Assembly languages
- High-Level languages

The **abstraction** level of programming languages from hardware is what distinguishes programming languages, with machine languages providing no abstraction, assembly languages providing less abstraction from the hardware, whereas high level languages provide a higher level of abstraction.



Source: https://codeforwin.org/2017/05/programming-languages-classification.html

Machine Language

Definition:

A computer programming languages consisting of binary or hexadecimal instructions which a computer can respond to directly and execute directly through it's CPU.

- Comprised of 1s and 0s
- The "native" language of a computer
- Difficult to program – one misplaced 1 or 0 will cause the program to fail

- Code example:

  1110100010101

  111010101110

  10111010110100

  10100011110111

Compiled vs. Interpreted Languages

Every program is a set of instructions, whether it's to add two numbers or send a request over the internet. Compilers and interpreters take human-readable code and convert it to computer-readable machine code. In a compiled language, the target machine directly translates the program. In an interpreted language, the source code is not directly translated by the target machine. Instead, a *different* program, aka the interpreter, reads and executes the code.

Definition:

Compiled languages *are converted directly into machine code that the processor can execute.*

- Tend to be faster and more efficient to execute than interpreted languages
- Give the developer more control over hardware aspects, like memory management and CPU usage.
- Need a "build" step - they need to be manually compiled first. You need to "rebuild" the program every time you need to make a change.

Examples of pure compiled languages are C, C++, Erlang, Haskell, Rust, and Go.

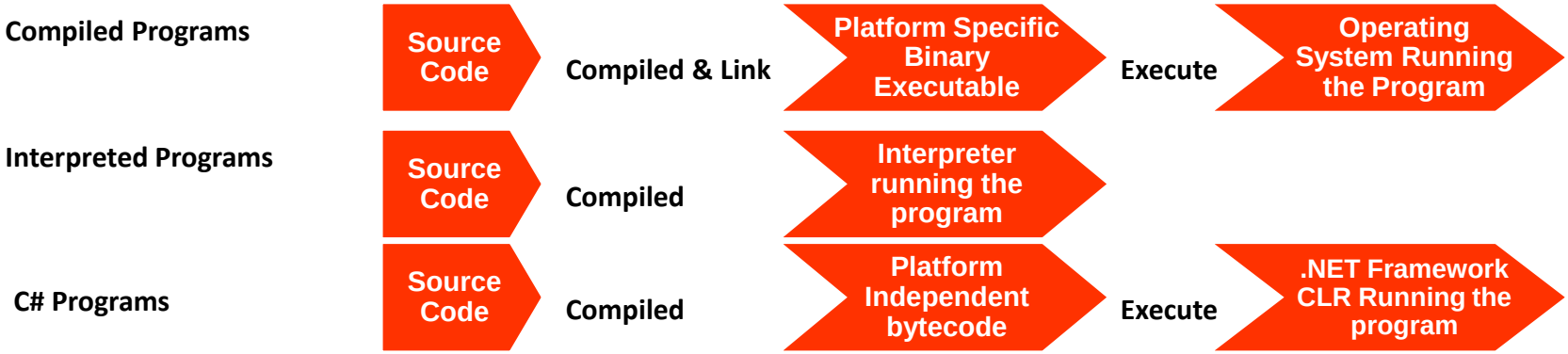
Definition:

Interpreters will run through a program line by line and execute each command.

- Interpreted languages were once known to be significantly slower than compiled languages. But, with the development of just-in-time compilation, that gap is shrinking.

Examples of common interpreted languages are PHP, Ruby, Python, and JavaScript.

Source: https://guide.freecodecamp.org/computer-science/compiled-versus-interpreted-languages/



Perform Static Analysis

Definition:

Static program analysis *is the analysis of computer software that is performed without actually executing programs*

- Analysis performed on executing programs is known as dynamic analysis).[
- In most cases the analysis is performed on some version of the source code, and in the other cases, some form of the object code.

Source: https://en.wikipedia.org/wiki/Static_program_analysis

UNIX (trademark)

Definition:

Unix systems *are characterized by a modular design that is sometimes called the "Unix philosophy": the operating system provides a set of simple tools that each performs a limited, well-defined function, with a unified filesystem (the Unix filesystem) as the main means of communication, and a shell scripting and command language (the Unix shell) to combine the tools to perform complex workflows.*

- Unix distinguishes itself from its predecessors as the first portable operating system: almost the entire operating system is written in the C programming language, thus allowing Unix to reach numerous platforms

Source: https://en.wikipedia.org/wiki/Unix

Unix Command Line and its Tools

- **Shell programming**
  - Command — Execute a simple command
  - Echo — Write arguments to standard output
  - Printf — Write formatted output
- **File System**
  - cd — Change the working directory
  - cp — Copy files
  - df — Report free disk space
  - ls — List directory contents
  - mkdir — Make directories

Source: XXXXX

Class Exercise

- Open your command prompt and try various commands
  - cd
  - dir
  - etc…

Build Tools

Definition:

*A **build tool** is a programming utility that is used when building a new version of a program.*

Examples:

- make is a popular open source build tool that uses makefile, another build tool, to ensure that source files that have been updated (and files that are dependent on them) will be compiled into a new version (build) of a program
- MSBuild: The build system for Microsoft and Visual Studio.

IDE (Integrated Development Environment)

Definition:

A software application that provides comprehensive facilities to computer programmers for software development

Consists of a:

- source code editor
- build automation tools and
- debugger

Source: https://en.wikipedia.org/wiki/Integrated_development_environment

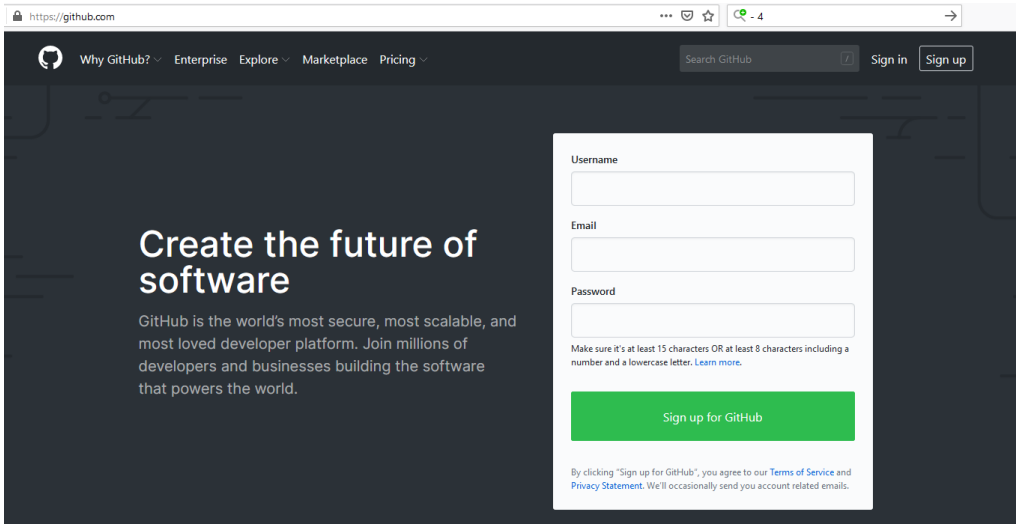
GitHub

GitHub is a web-based Git repository hosting service.
https://github.com/

- Offers all of the distributed version control and source code management (SCM) functionality.
- Provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.
- Why use GitHub? You are going to upload and edit your exercises on your personal GitHub page

Let's Sign up!

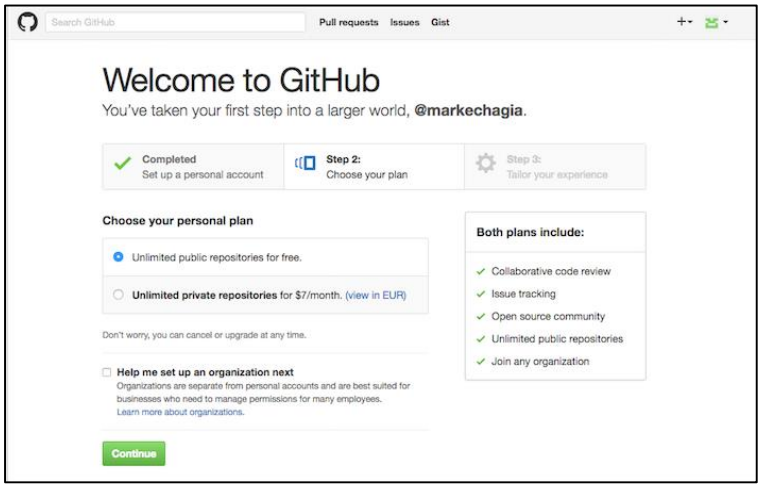
1. Visit github.com
2. Type in your **Username**
3. Type in your **Email**
4. Type in your **Password**
5. Press **Sign up for GitHub**
6. If requested, verify your are a person by solving the puzzle



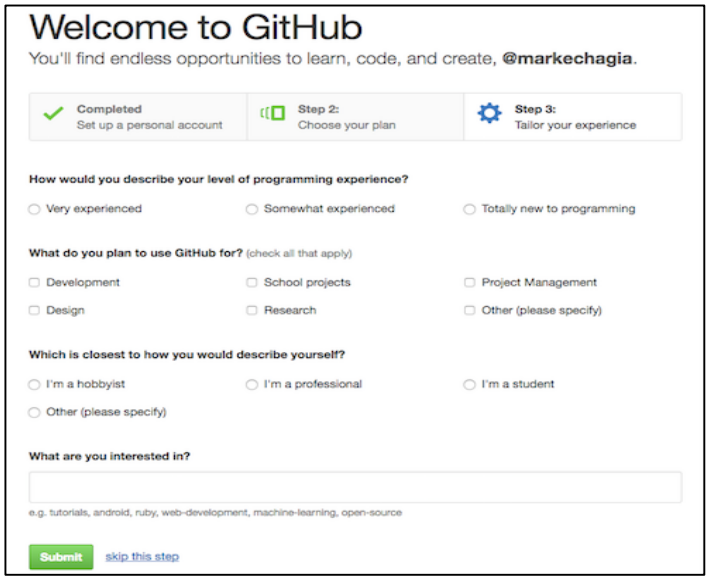
All images captured for educational purposes from github.com (February 2020)

Sign Up for GitHub: Create Your Profile

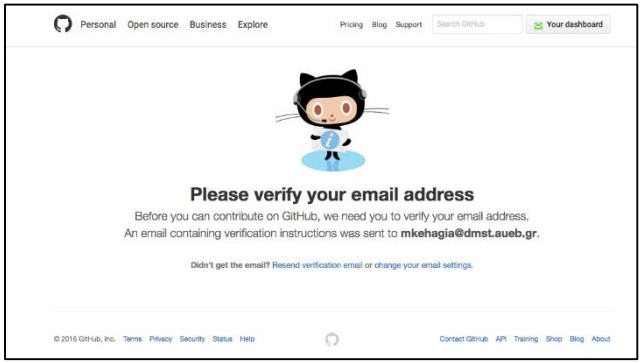
7. Select the **FREE** repositories option and then press **Continue**



8. Answer the **Welcome** questions or **Skip this step**



9. Verify your email (visit your email and click the link)
10. Now you are **Ready to go!**

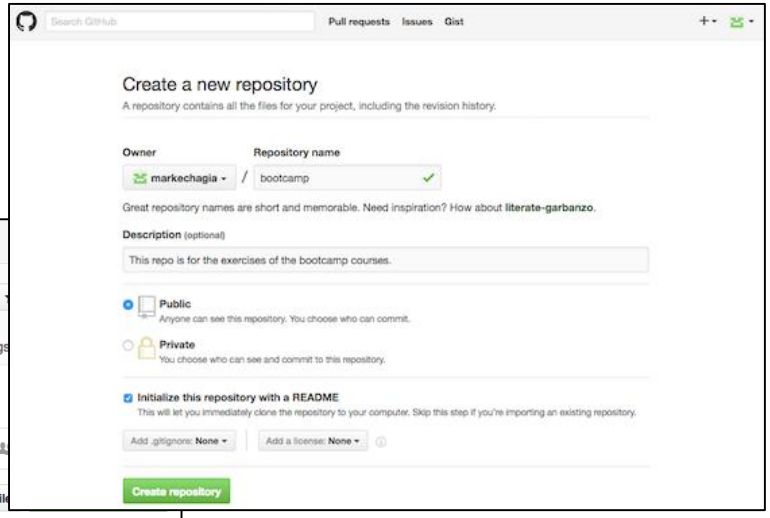


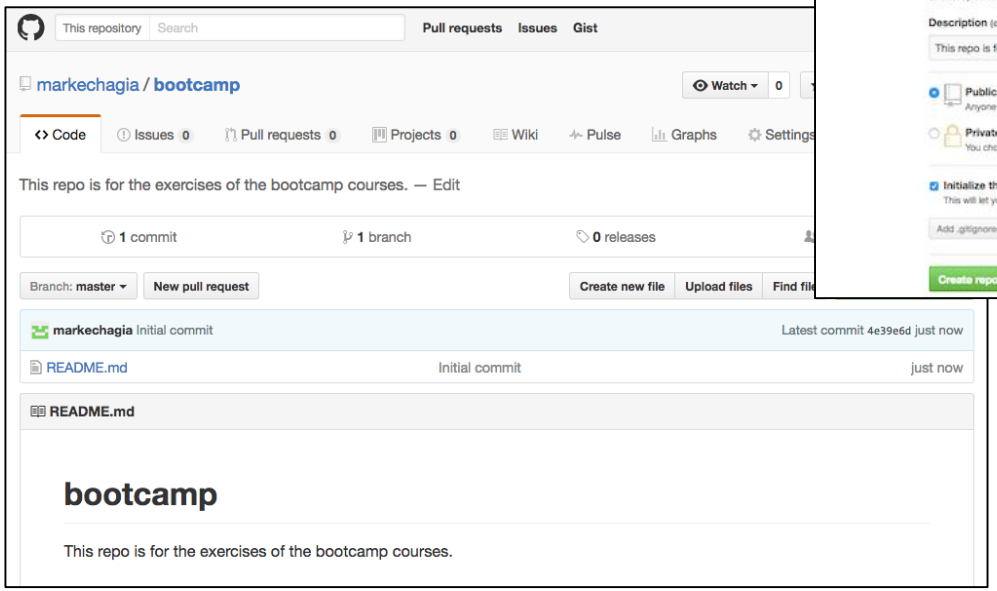
All images captured for educational purposes from github.com (February 2020)

Create a Repo

1. Start a Project

2. Give a name to your repo
3. Write a description for your repo
4. Select to be public
5. Select to initialize the repo with a README file

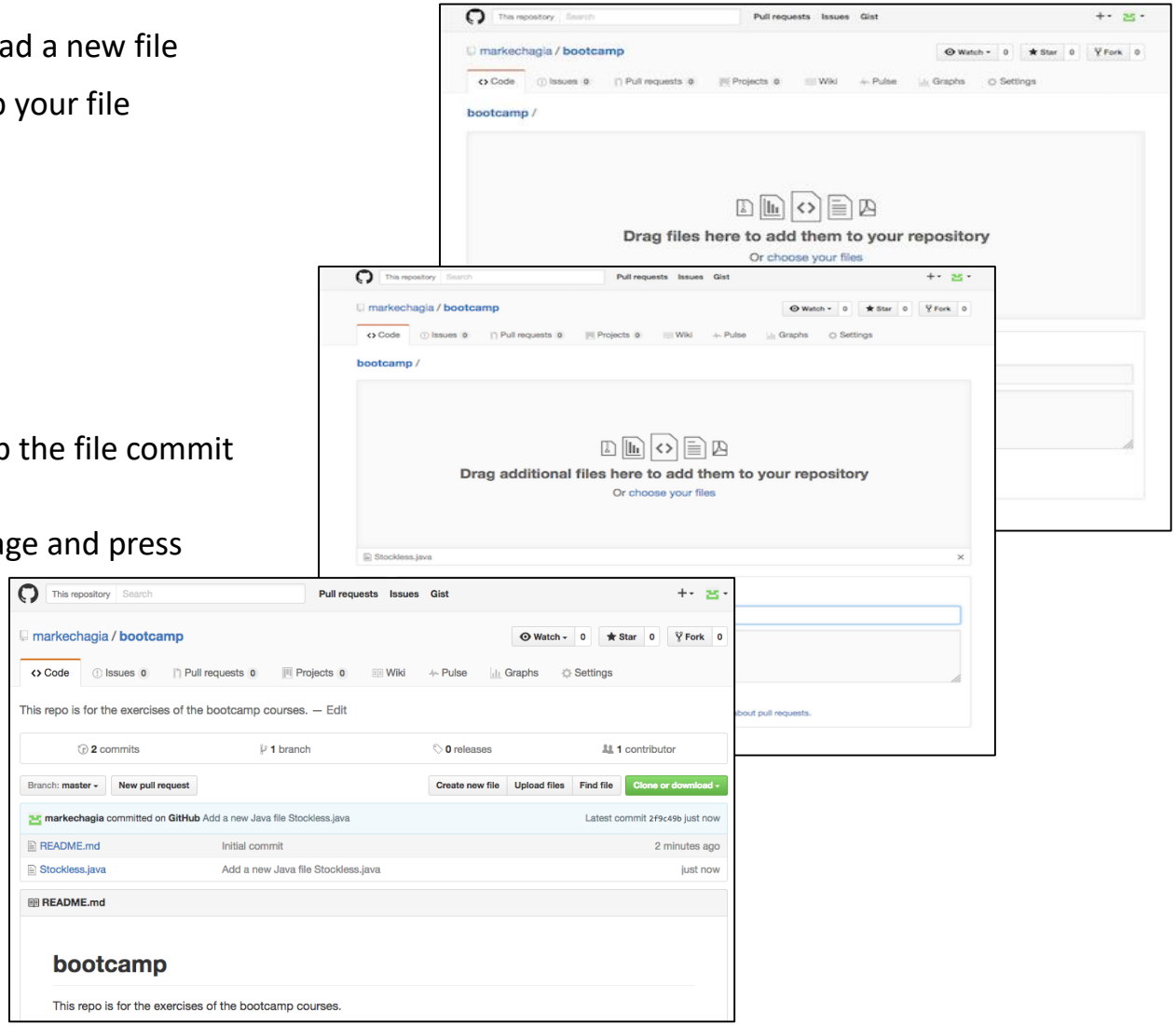




All images captured for educational purposes from github.com (February 2020)

Add a New File to the Repo

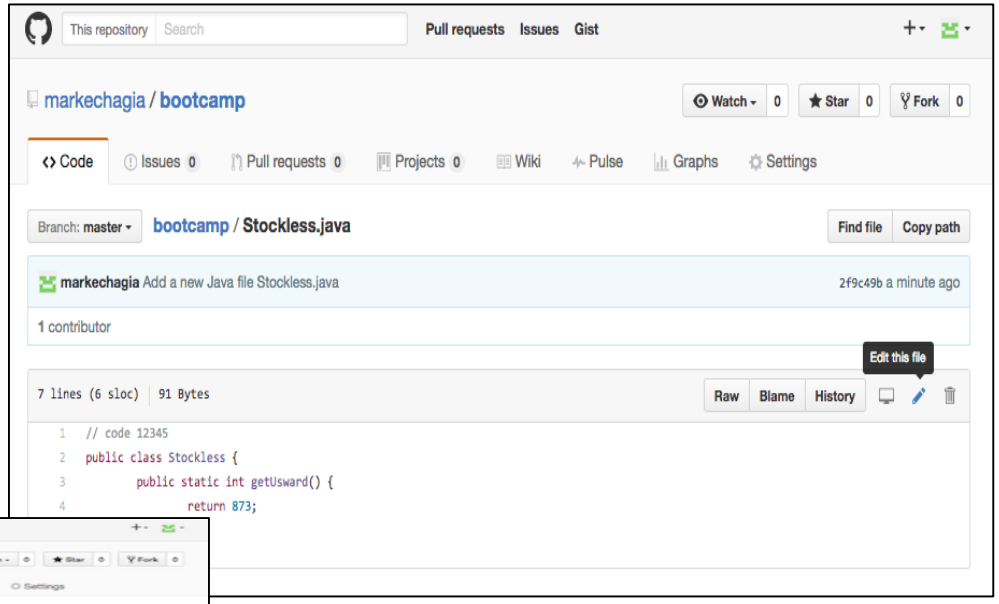
1. Select to upload a new file
2. Drag and drop your file
3. After you drop the file commit your changes
4. Write a message and press **Commit**

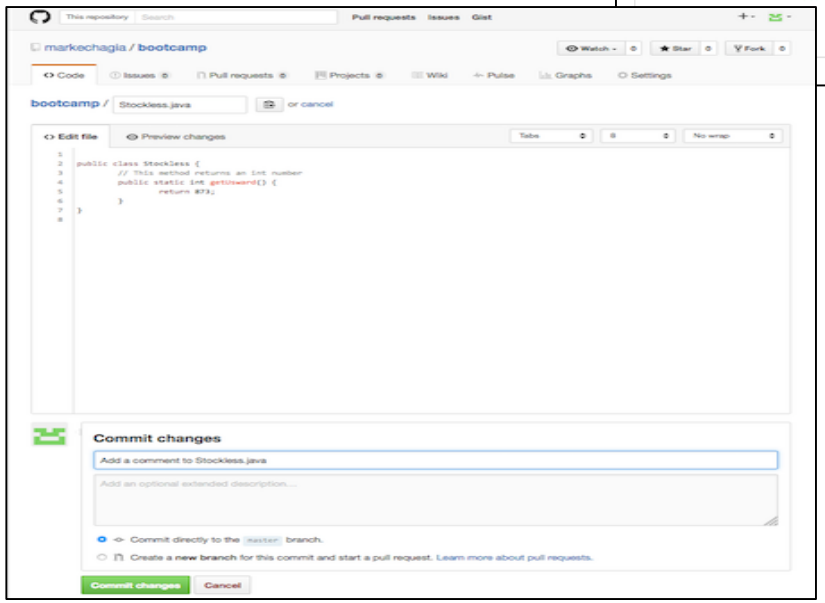


All images captured for educational purposes from github.com (February 2020)

Edit a File

1. Select a file and open it
2. Make any changes you want
3. Commit your changes

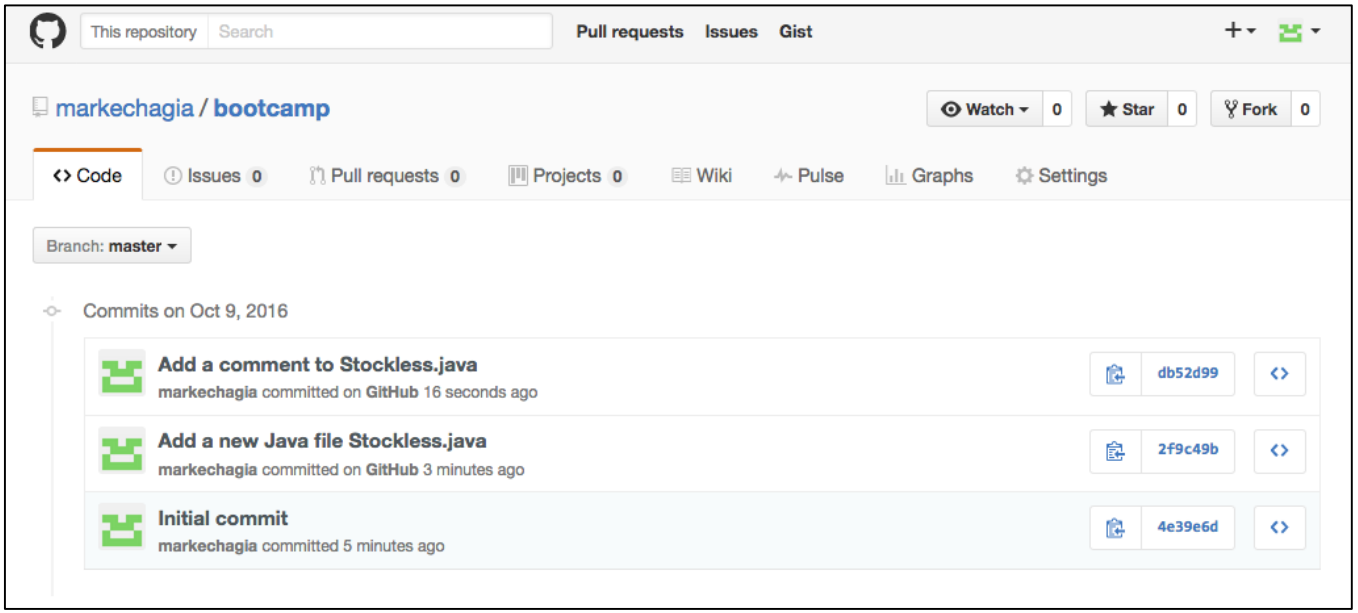




All images captured for educational purposes from github.com (February 2020)

Repo Files

- This is what a usual GitHub looks like!



Class Activity

- Create your own GitHub account and experiment with some sample files!

All images captured for educational purposes from github.com (February 2020)

It’s time for a Knowledge Check!

Sample Questions

1. A front-end application which include GUI is created using _____.
   A. Compiler
   B. IDE
   C. Interpreter
   D. Command line

2. What does the **sleep** command (UNIX Command line) line?
   A. Write arguments to standard output
   B. Evaluate arguments as an expression
   C. Execute a simple command
   D. Suspend execution for an interval

3. What is GitHub?
   A. A control repository hosting service
   B. An IDE
   C. A Compiler
   D. A Static Analysis tool for code review

Sample Questions

1. A front-end application which include GUI is created using _____.
   A. Compiler
   B. IDE
   C. Interpreter
   D. Command line

2. What does the **sleep** command (UNIX Command line) line?
   A. Write arguments to standard output
   B. Evaluate arguments as an expression
   C. Execute a simple command
   D. Suspend execution for an interval

3. What is GitHub?
   A. A control repository hosting service
   B. An IDE
   C. A Compiler
   D. A Static Analysis tool for code review

Coding Bootcamp

Introduction to Programming

Objectives	Syllabus Topics
▪ Types of programming languages ▪ Front end & Back end development	▪ 2.2 Programming Languages and Paradigms

Category	Topic	Task
FSD_2 Introduction to Programming	**2.2 Programming Languages and Paradigms**	2.2.1 List the different types of programming languages like: procedural, object-oriented, data oriented, scripting. Distinguish between compiled and interpreted languages
		2.2.2 Distinguish between back-end and front-end software development

- ✓ Understand and list the different types of computer programming languages
- ✓ Understand what are procedural programming languages
- ✓ Understand what are object oriented programming languages
- ✓ Understand what are data oriented programming languages
- ✓ Understand what are scripting languages
- ✓ Differentiate between compiled and interpreted languages
- ✓ Know examples of all of these types of languages
- ✓ Know what back end software development is
- ✓ Know what front end software development is
- ✓ Understand the difference and the connection between front end and back end software development

Different Types of Programming Languages

- Procedural
- Object-oriented
- Data oriented
- Scripting
- Compiled and interpreted languages

Procedural Languages

Definition:

*A **procedural language** is a computer programming language that follows, in order, a set of commands.*

- Procedural programming languages are based on the concept of the unit and scope (the data viewing range of an executable code statement).
- They use functions, conditional statements and variables to create programs from which the computer displays output, after appropriate calculations are executed
- A procedural program is composed of one or more units or modules, either user coded or provided in a code library; each module is composed of one or more procedures, also called a function, routine, subroutine, or method, depending on the language

Examples: COBOL, Pascal, Java, C, Python, Fortran

Source: https://www.computerhope.com/jargon/p/proclang.htm

Object Oriented Languages

Definition

An Object-oriented language (OOL) is a high-level computer programming language that implements objects and their associated procedures within the programming context to create software programs.

- Object-oriented programming (OOP) is a programming paradigm based on:
  - the concept of "objects”
  - code, in the form of procedures, often known as methods
  - binding related data and functions into an object
  - reuse of objects within the same and/or other programs
- In OOP, computer programs are designed by making them out of objects that interact with one another.
- Most popular languages are class-based, meaning that objects are instances of classes, which typically also determine their type.

Examples: C++, Object Pascal, Java, etc.

Sources: https://en.wikipedia.org/wiki/Object-oriented_programming
https://www.techopedia.com/definition/8678/object-oriented-language-ool

Scripting Languages

- A ***Scripting language*** has two apparently different, but in fact similar, meanings.
  - In a traditional sense, scripting languages are designed to automate frequently used tasks that usually involve calling or passing commands to external programs. Many complex application programs provide built-in languages that let users automate tasks. Those that are interpretive are often called scripting languages.
  - Recently, many applications have built-in traditional scripting languages, such as Perl or Visual Basic, but there are quite a few native scripting languages still in use.

Source: https://en.wikipedia.org/wiki/Scripting_language

Interpreted Languages

- An **interpreted** language is a programming language for which most of its implementations execute instructions directly, without previously compiling a program into machine-language instructions.
- The interpreter executes the program directly, translating each statement into a sequence of one or more subroutines already compiled into machine code.

Data Oriented Languages

- Data-oriented languages provide powerful ways of searching and manipulating the relations that have been described as entity relationship tables which map one set of things into other sets.
- Examples of data-oriented languages include:
  - SQL
  - dBase
  - Clipper
  - Clarion

https://en.wikipedia.org/wiki/List_of_programming_languages_by_type#Data-oriented_languages

Compiled Languages

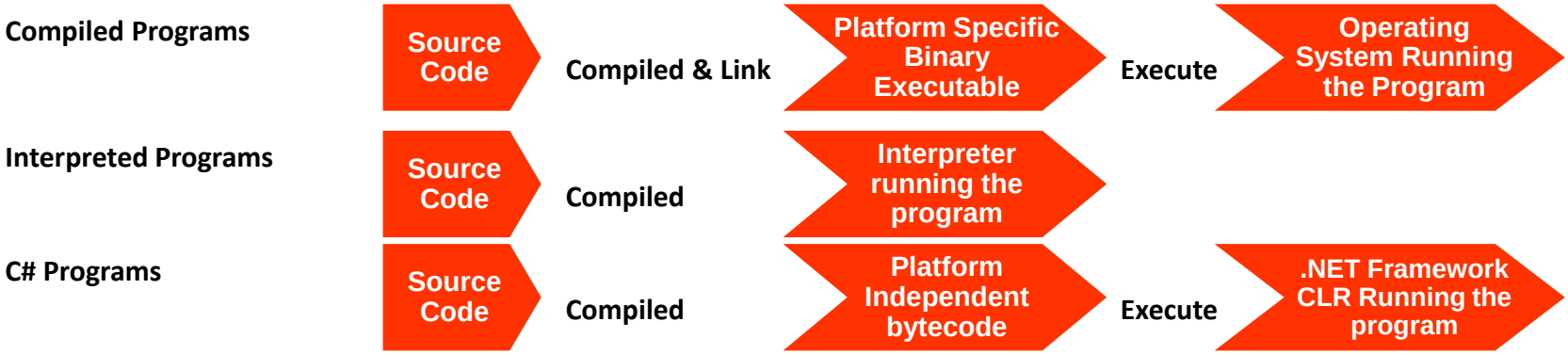
- A **compiled language** is a programming language whose implementations are typically compilers (translators that generate machine code from source code), and not interpreters (step-by-step executors of source code, where no pre-runtime translation takes place).

Compiled vs Interpreted Languages

- In principle, any language can be implemented with a compiler or with an interpreter.
- A combination of both solutions is also common: a compiler can translate the source code into some intermediate form (often called p-code or bytecode), which is then passed to an interpreter which executes it.

https://www.codeproject.com/articles/696764/differences-between-compiled-and-interpreted-langu

Remember this from a few slides before?



Back-end and Front-end Software Development

- Refers to the:
  - separation of concerns between the presentation layer (front end), and the data access layer (back end) of a piece of software
  - physical infrastructure or hardware.

Example: Client–server model

- the client is considered the front end
- The server is usually considered the back end

Source: https://en.wikipedia.org/wiki/Front_and_back_ends

Back-end & Front-end Software Development

In software engineering, the terms **front end** and **back end** refer to the separation of concerns between the presentation layer (front end), and the data access layer (back end) of a piece of software, or the physical infrastructure or hardware.

Back end Software development:

- Code that runs on the server
- Sending meaningful data to the frontend
- Interacts with the database
- Popular backend languages include PHP, Java, C#, Python and Ruby
  - the backend is responsible for saving user to the database.
  - when saved the backend can then fetch, update and delete this user

Front end Software development:

- Code that runs in the browser
- Structure the data that was sent from the backend
- In order to make it pretty we use CSS
- In order to make it interactive we use JavaScript
- Combine HTML, CSS and JavaScript — collectively known as the frontend — create the User Interface (UI)

Example: The look on your Twitter account and DM interactivity

- when choosing how your account looks, you are using CSS
- when a messaging modal/pop-up appears when another user's name in is clicked, JavaScript is used to add that interactivity

Source: https://en.wikipedia.org/wiki/Front_and_back_ends

The Database

Definition

A database is an organized collection of structured information, or data, typically stored electronically in a computer system.

A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.

Usually a database (DB):

- has columns and rows
- contains all the important information that your website needs to remember
- Stores information

Example: Your latest tweet on your Twitter account

- you tweeted on your Twitter timeline
- you navigated away from the site
- when you return to twitter your tweet is on your timeline because it was saved to the database

Source: https://www.oracle.com/database/what-is-database.html

Class Exercise

- Discuss with students popular web applications, like social media in order to understand how they work; distinguish between their front-end, their back-end and how data is stored and retrieved.

It’s time for a Knowledge Check!

Sample Questions

1. Which of the following is an interpreted programming language?
   A. Delphi
   B. ADA
   C. C#
   D. JavaScript

2. Which of the following languages is NOT usually used for Back-End Software development?
   A. PHP
   B. Java
   C. JavaScript
   D. C#

3. Suppose you log on your Facebook account and look for and locate your last post. Where is that post actually stored?
   A. Facebook's database
   B. Facebook's front end
   C. Facebook's back end
   D. On your computer's drive

Sample Questions – Answers

1. Which of the following is an interpreted programming language?
   A. Delphi
   B. ADA
   C. C#
   D. JavaScript

2. Which of the following languages is **NOT** usually used for Back-End Software development?
   A. PHP
   B. Java
   C. JavaScript
   D. C#

3. Suppose you log on your Facebook account and look for and locate your last post. Where is that post actually stored?
   A. Facebook's database
   B. Facebook's front end
   C. Facebook's back end
   D. On your computer's drive

Any Questions?

References | Further Reading

- List References

PeopleCert Values Your Feedback

Like the course?
Have something to say?

Send us your comments at:
academic@peoplecert.org

Thank You!

For latest news and updates follow us.

linkedin.com/company/peoplecert-group/
twitter.com/peoplecert
youtube.com/channel/UCLBidKZS9Xk08f_PM5Edtfg
facebook.com/peoplecert.org