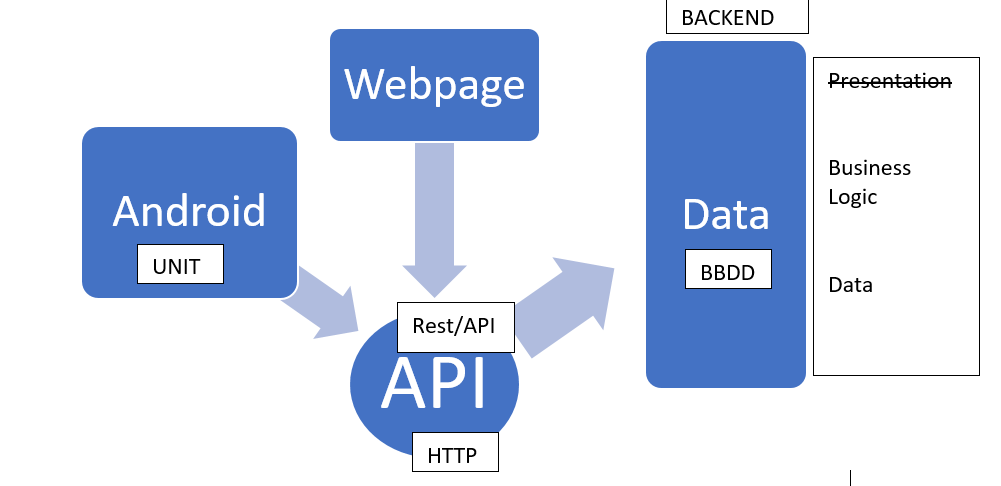
Points to take in account

1. Management
2. Tools 🡪
   1. IDE (INTELLIJ,ANDROID STUDIO)
   2. CVS,CVN,GIT
   3. TDD 🡪 JUNIT
3. OOP

From test to Development!

The core of the project is the backend where our Core Data is stored and can only be accessed via an API (RESTFULL API) which is communicated in HTML language.

Both of the Frontend Webpage and Android Application Unity communicates with Core Data via the API aforementioned.



Evaluation

Min 1 🡪 27/03/2020

Min 2 🡪 29/05/2020

Final Presentation of the Project 23/06/2020

Min 1 🡪 Everything Backend and API

Comment

First Assignment

IntelliJ & Install GitHub, to do this go to a random Java Project and download it via GitHub to edit on the PC.

Office Hours🡪 & Via Mail antoni.oller@upc.edu

Juan Lopez

Correo: [lopez@ac.upc.edu](mailto:lopez@ac.upc.edu)

Despacho: 018

Teoria Actualidad

Passar de varios protocolos a un solo protocolo para todos los servicios.

Ejemplo

s

Utilizamos XML/JSON por que permiten hacer envío de archivo sin ningún formato que hace que sea implementado y modificable en cualquier sitio.

JavaScript fue implementado por comprobar correos pero a partir de allí empezó el despego de JavaScript en ser utilizar todos los sitios. Hasta empezó ser útil para guardar la pagina web en el cliente para que la próxima vez que ejecute la pagina web le abra mas rápida y solo tenga que actualizar partes de paginas WEB que fueron anadidos y modificados.

Estos tipos de servicios HTTP 1 son Petición y Respuesta, pero por eso fue desarrollada nueva version de HTTP 2 que permite comunicación bidirectional.

La nueva versión HTTP 3 esta implementando UDP y no TCP porque es mucho más eficiente ya que no hay RTTX.

27/02/2020

|  |  |  |
| --- | --- | --- |
| OO | JAVA | UML |
| ENCAPSULACION | INTERFACES | DIAGRAMA DE CLASES |
| CLASE | CLASES (Constructor, Métodos, Atributos) |  |
| INSTANCIA/OBJETO | CLASES ABSTRACTAS |  |
| RESP. INSTANCIA | STATIC |  |
| RESP. CLASE | PROTECTED/PUBLIC/PRIVATE |  |
| HERENCIA |  |  |
| POLIMORFISMO |  |  |

ENCAPSULACION: Encapsulación de datos y funciones.

Instancia: Instancia es un objeto creado para mantener unos datos o clases.

Responsabilidad Instancia: Cada instancia mantiene su carácter especial o datos que diferencian de otras instancias.

UML: Nos permite hacer fotogramas o grafica de nuestro programa que ayuda a verlo más fácilmente, es decir se puede ver las relaciones entre varios objetos, entidades y como son esas relaciones entre ellos.

STUDENT

-name: String

-surname: String

+m (): void

=m2 (a): int

#m3 (): void

Los signos(+,-,# sirven para indicar si son private, public o protected)

Package: Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces. Packages are used for:

Preventing naming conflicts. For example, there can be two classes with name Employee in two packages, college.staff.cse.Employe and college.staff.cse.Employe

Making searching/locating and usage of classes, interfaces, enumerations and annotations easier

Providing controlled access: protected and default have package level access control. A protected member is accessible by classes in the same package and its subclasses. A default member (without any access specifier) is accessible by classes in the same package only.

Packages can be considered as data encapsulation (or data-hiding).

Example: Student Miguel = new Student (Miguel,...)

Responsibilidad Clase: Mantiene las clases por ejemplo los tipos de objetos y como de visible son dentro de ella (static, private, public, protected).

Ejemplo InteliJ🡪 A partir de UML hacer el programa.

**Próxima clase**

Colecciones

List, Hash Map, …

Vector, ...

Arrays 🡪 JAVADOC

SORT(OBJETOS)

Lo que será explicado es el orden de criterio natural, tiene que implementar un interfaz en nuestro caso comparable en figura

------------------------------------------------------03/03/2020---------------------------------------------------------

**Bootstrap:** To create responsive apps on the web. To use the Bootstrap in HTML is by using <INCLUDE>. All the files are finished with Bootstrap extension min.css. The min files are normal files minimized in size to use them for lower use of memory. MaxCDN where the stylesheet, jQuery library and the JavaScript file for to use on the Web Page.

Remember to reload on pages using shift + reload to have the un-cached page or force reset of the page.

Page Tutorial Bootstrap🡪https://www.w3schools.com/bootstrap/default.asp

JQuery 🡪 https://www.w3schools.com/jquery/default.asp

https://api.jquery.com/jQuery.get/

JQuery with JSON example: https://stackoverflow.com/questions/8951810/how-to-parse-json-data-with-jquery-javascript

GitHub API V3: https://developer.github.com/v3/

Bootstrap only allows the webpage to be divided in grids of up to 12 units.

CSS means cascaded style sheets.

JQuery is used to modify and access the elements of the HTML. Adding actions to the document HTML.

JavaScript works on the Selector ($) which selects the code to run using jQuery.

JQuery AJAX: Petition from client to server, asynchronously element by element.

**HOMEWORK ATENEA – USEFULL INFORMATION (INDIVIDUAL)**

URLS-USEABLE in GITHUB for good looking JSON (JSON VIEWVER)🡪 api.github.com 🡪 api.github.com/users/jlopezr && api.github.com/users/jlopezr/repos

En las interfaces se especifica qué se debe hacer pero no su implementación.

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| 1) Abstract class can **have abstract and non-abstract** methods. | Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also. |
| 2) Abstract class **doesn't support multiple inheritance**. | Interface **supports multiple inheritance**. |
| 3) Abstract class **can have final, non-final, static and non-static variables**. | Interface has **only static and final variables**. |
| 4) Abstract class **can provide the implementation of interface**. | Interface **can't provide the implementation of abstract class**. |
| 5) The **abstract keyword** is used to declare abstract class. | The **interface keyword** is used to declare interface. |
| 6) An **abstract class** can extend another Java class and implement multiple Java interfaces. | An **interface** can extend another Java interface only. |
| 7) An **abstract class** can be extended using keyword "extends". | An **interface** can be implemented using keyword "implements". |
| 8) A Java **abstract class** can have class members like private, protected, etc. | Members of a Java interface are public by default. |
| 9)**Example:** public abstract class Shape{ public abstract void draw(); } | **Example:** public interface Drawable{ void draw(); } |

General Difference between interface and class, interface implents while

Implement Interface.

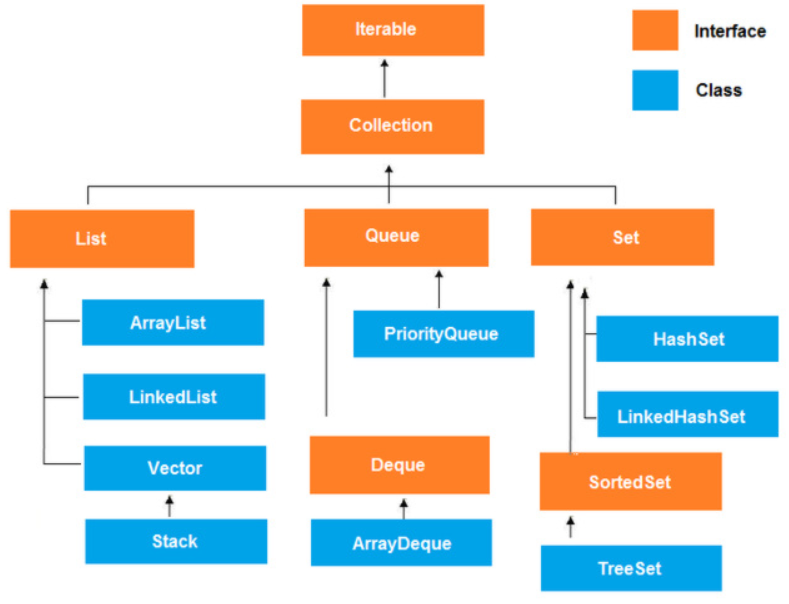
|  |  |
| --- | --- |
| Enlaces GITHUB | Herramientas |
| +Interfaces | Maven |
| +Tipos Genéricos | Junit |
| Estructura de Datos  -Listas 🡪 Basadas en Arrays   * + - Nodos Encadenados     - Tablas de Hash     - Colas/Pilas | LOG4J |
| +Clases Abstractas |  |

Big O(n) notation:

What is the cost of using a certain algorithm, and how it does not depend on the type of machine but a mathematical theory?

O(1),O(N)[Añadir un element en lista con puntero], O(N^2)[Busqueda Elemento con elemento],O(log2N)[Arboles Equilibrados definidos para listas].

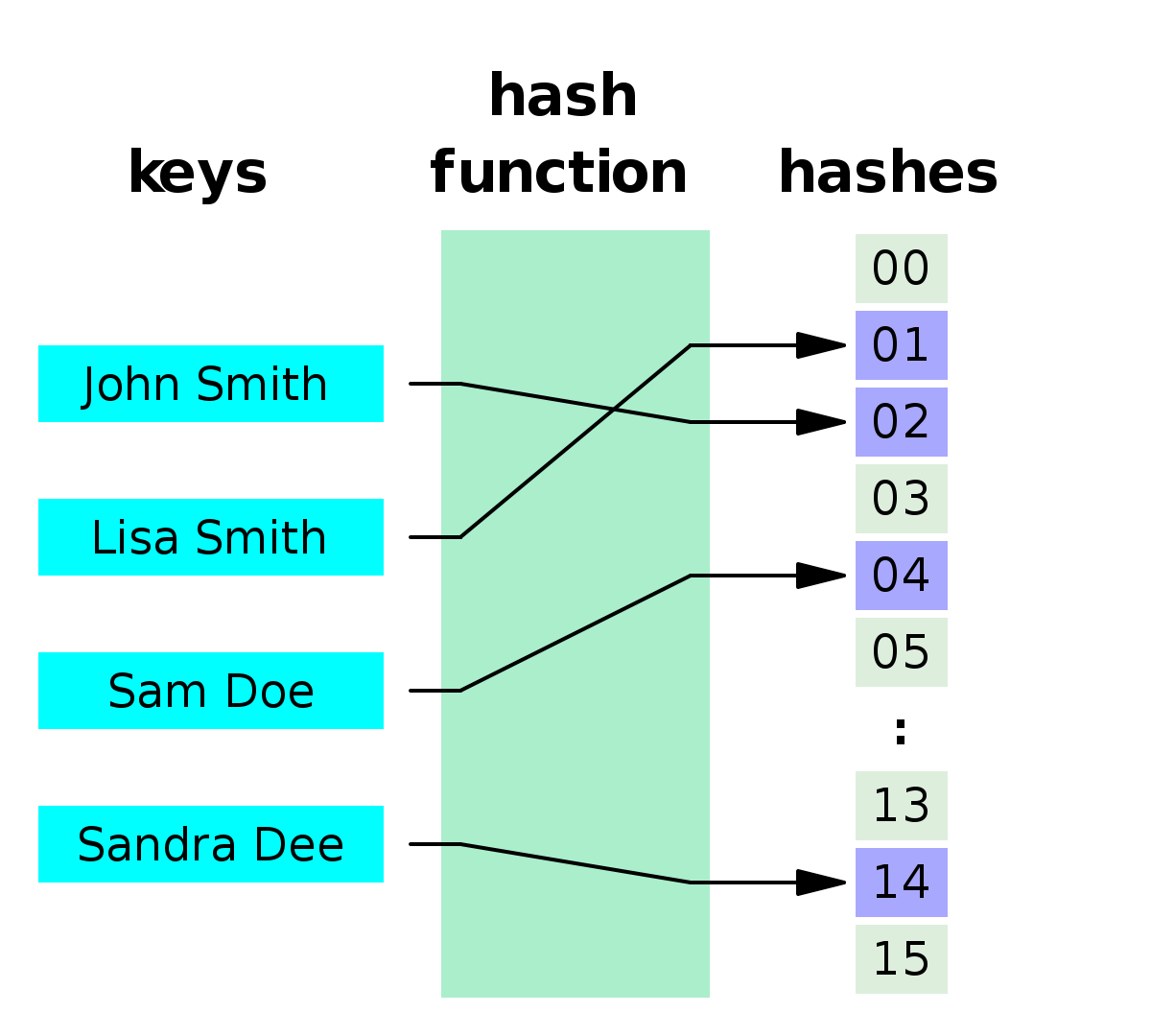
Hash Table🡪 Theoretical O(constant) en realidad es (NLogN).

Vamos a utilitzar listas que son proporcionados por coleciones de Java.

Listas tienen un coste O(constante) para añadir un elemento despues de ultimo elemento de lista ya que tenemos un punter, lo mismo ocurre cuando queremos añadir un elemento en medio de la lista ya que solo desconectamos el enlace link entre elemento y añadir nuevo link.

Vector tiene un coste de buscar un elemento O(lineal), pero encambio si ya lo tenemos ordenado el coste passa a ser una busqueda iscotonicas.

Una Tabla de Hash es un vector Horitzontal, que es un simple vector mas una funcion. Esta tambien se le denomina funcion de Hash o funcion de dispersion. No es reversible es decir en una direccion no hay problema y siempre devuelve el mismo Hash para mismo valor pero no es possible recuperar la informacion original.





Ideal Hash Conversion O(N) Real Hash Collisions O(nLogN)

Problemas de Hash esque depende de la funcion de dispersion que puede crear problemas de collision y baja de O(N) a O(NLogN).

Deberes 🡪 Papel