M. N. IS. Alex and I choose this career because I like programming computers and If I have fun with this, why don't be this my job. And I learn about the introduction of programming, one C language, like different kind of signal hardware

USC → Github.

JRE → JKD → IDE

02 Pry. Descrip.

Artefactos – Analisis – Diseño – Requerimientos – Entrevistas, etc.

Git

* Use hitGub

Clone → Solo
Pull → Solo

Deber: Entrevista Usuario.
→ Nombre Usuario:
→ Empresa / Sitio
→ Contactos: Hf, correo
→ Cargo:

git + add -a → all.

git commit -m " "

Hidden Figures
The imitation game

JAVA : NEED
→ JRE → JVM
→ JDK
→ Net Beans (IDE)

Norma

Soy ambientalmente consciente: uso papel de caña.

Papel de caña de azúcar

UFML 1.2 1-3 3-5 3-5

DD MM AA

Java → Sun Micro Systems (Old)
→ Oracle (New)

EDD → 1-2

Aprender a leer datos en java
→ Pedir usuario la tabla que desea

// What multiplication table do you want to see (for)
// enter multi a number
// if the number is even or odd (if)
// ask again for the table number and even/odd and exit (while)

Lunes

Imperativo OPTIONS OOP, PP. Pascal, NODE, JAVA, ETC

Imperativo → Como proveer

Declarativo → Qué quiero que haga

Concepts OO

Encapsulado → Palabra reservada → Private

Abstraido → Palabra public

Classification

Private: Public (Abstracción)
Private (Encapsulamiento)
Clase

Clase → El singular

REFACTORING → CHANGE IDENTIFICATORS

DEBER: Objetos → Proyectos
UFML.

Clases → Mayúsculas
Objetos, variables, métodos → Minusculas

Camel Case ⟷ myChicken

Atributos (-)
Métodos (+ o -)

DD MM AA

Proyect.
1. List objects
2. Attributs and operations for the Objects
3. Model using paper and pen
4. Create classes in Java - Netbeans.

WSDQ UML ; WSTO Associations
WST2 From UML to Code → UML Edcor site

snake_case
cammel Case

Capital (TODO MAYUS)
lowel camel Case — small case
Uper cammel Case
Lowe case
snake case
uper case
upper snake_case

Ctr-f (buscador)

------> Depende de

------◇ Agregado de

_____ Asociations (more hard than dependence)
Contiene objetos

- Diseño siempre en tercera persona.
- Asociacion → Relacion atributo de una clase
- Dependencia → Relacion funcion de una clase

- Constructor → método → Crea un objeto con valores iniciales

Por defecto Vacio Con parametros

INSPECTIONS and WALK THROUGH

setter
getter
> METODOS QUE SACAN INFORMACION

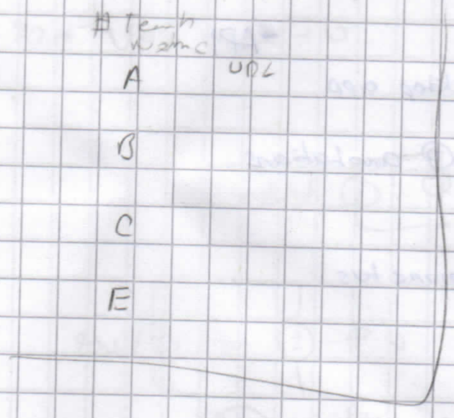

Diagrama - Nombres Clases.
- Atributos (visto)
- Metodos Nelhos

→ Se baja por global no por errores

Ws bnec →

Agregations → (include) (contains) Subpart

Recursive Agregation Agregate the same class

→ Reflexitive - Transitive
→ Asymmetrical

Composition → Ciclo de vida STRONG

Case Diagram

Actors ← Users, Other system, Hardware

- Begin with verb

Use case answer what and Why

Inheritance → General

Substitubility

Files . .csv Comma separate value
.xml Extensible markup language
.Json Java script notation

Exceptions Try catch
Inheritance extends
Polimorphysm @Override

- Json libreria (Gson).

USER — PASWORD | Contacts Book

Unit test *API
GUI → Grafical user interface → Desktop app.

Polymorphism: - poly 'morphic Objets @ anotations
- Overriding ( );
- Overloading ( ) {...}
Type of paramethers / number of parameters

Data Base in cloud
- Acces throught the cloud
- Company should not buy hardware

Mongo DB --- Atlas
Free storage for everyone

Insertar datos
Leer datos } nube
Remover datos

Ver Mongo shell
Buscar MgDB Compass
Aplicación

Mecanicos - Tubos de vacío - Transistores - Chips

- Tecnicas
- Cuando
- Quien

- Método
- Caja negra
- Small

- Tester
- Quality asurance
-

Validations → Es hacerlo lo que debe hacer

Verifications → El que lo hago bien

Test → Configure System Setup
→ Stimulate
→ Observe

Test Data
¿Fuentes de falla?
¿Datos seleccionados para mostrar fallas?
¿Qué y cuántos pruebas?

Condiciones Java
- Every line
→ Every posible branch
→ Boundary conditions

- Failures
→ Bugs

Persistance Interface — WS 23 — Persistence Lib

DD MM AA

Interfaces - Interface

- Implements

No instanciar Interfaces

S Single responsability principle
O Open / close principle
L Liskov Sustitution principle
I Interface Segregation principle
D Depedency Inversion principle.

Design problem

S
- Every class be responsible for a single part of system's functionality
- A class's be encapsulate by the class
- A class's be narrowly aligned.

- Cohesion - Localization of designs decisions - Encapsulation

O
- Class open if il still abaible for extension
- Class close for use by other class

- Inheritance - Agregation - Parameterization

L if S is a seperalization of T, then an S object can be user whenever a T object is requiered

I
- No client be forced to depend on methods that it does not use.
- The public methods of a component can be grouped by purpose or responsability as captured in a declation in interfaces, or abstract classes.

D
- Organize system into layers
- Components from the alstract lasers should no depends on componets from the detail layers.
- Abstraction not depends on details
- Implementation details depend on abstractions

- Test → verification → right
- Unit test → every functionality (methods())

Syntax errors → int a

Test Cases

Add(x,y)	addend1	addend2	Expect Result	Actual Result
	2.0	3.0	5.0	
	3.2	4.6	7.8	
	0	-24.5	-24.5	
	-23.0	-45.1	-68.1	
	34000.23	60340.25	94340.48	

Class Abstracta → Funciones con cuerpo
→ Atributos

Interfaces → Terminan con i
→ Declara métodos

Persistence → Objects → Array List
GUI, DB → Data → ArrayList<Objects> → Model → Controller → View

Json connection mongo DB → Constructor
Read from Json to mongo DB

Quality Control → Project Quality
→ User - Acceptance Test plan
→ System Test plan

Test plans

- What to test?
- What techniques to use?
- How will the test cases be selected?
- What does testing start?
- When will it finish?

J Unit → library.
- Recibo un parámetro
- Función retorna algo

Errores de lógica

Programa que corre el programa (Automatización)

Fallas · Corregir errores · Corregir código

Patrón ⇒ Solución reusable

Patrón → Design choices by explaining consequences of known solutions

Gang of four 23 Patterns

Creational Patterns	Structural Patterns	Behavior Patterns
Single tone	Template	Strategy

- Introduction to Design Patterns
- 2.02.00 Parc
- Design Patterns Pt. 1.
- Design patterns Pt 2.