UTN SANTA FE

Diseño e Implementación

De Estructuras de Datos

Guía 06 – Excepciones - ENTREGABLE

Contenido

Fecha de entrega: 20 de Abril

Modalidad de entrega: URL de repositorio GIT

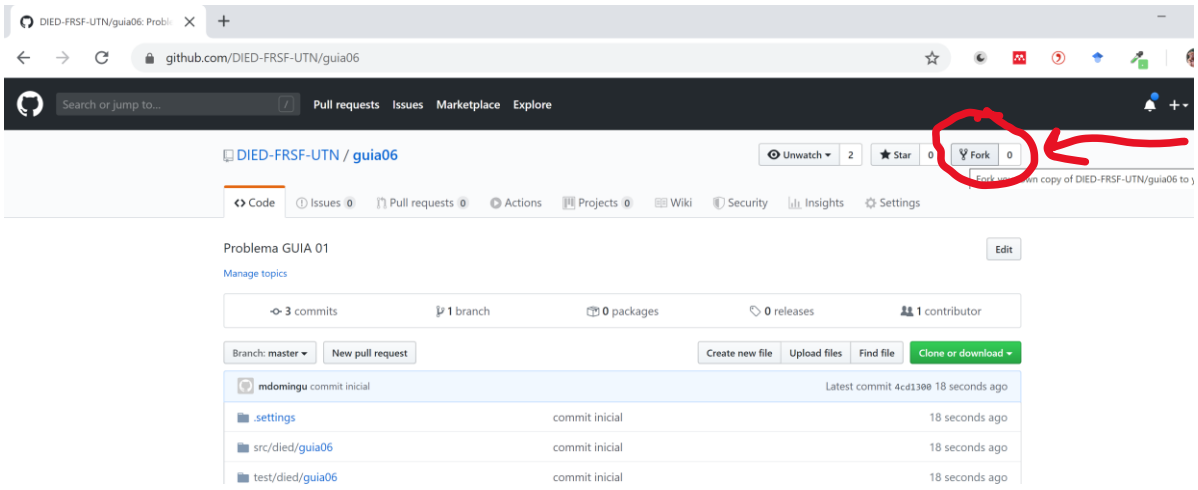
Guía 06 – Excepciones – Test Unitario - Comparaciones

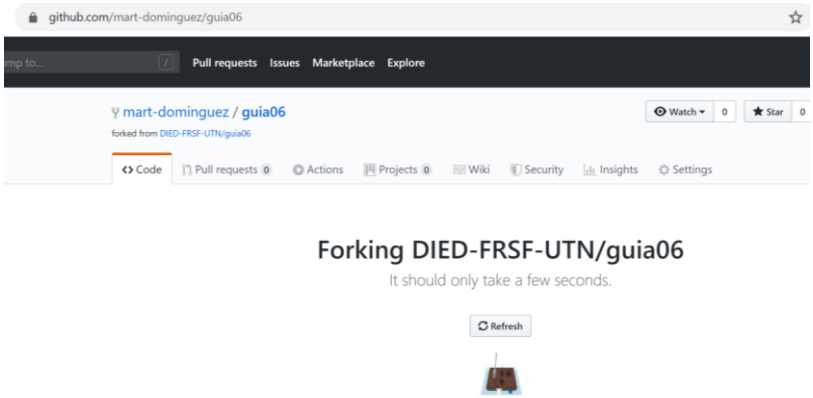
Se entrega un proyecto iniciado en ECLIPSE, en el siguiente repositorio GitHub.

Deberá realizar un "fork" del mismo (esto significa copiar el proyecto que se le entrega al espacio del repositorio de su usuario de github) y luego importarlo en eclipse.

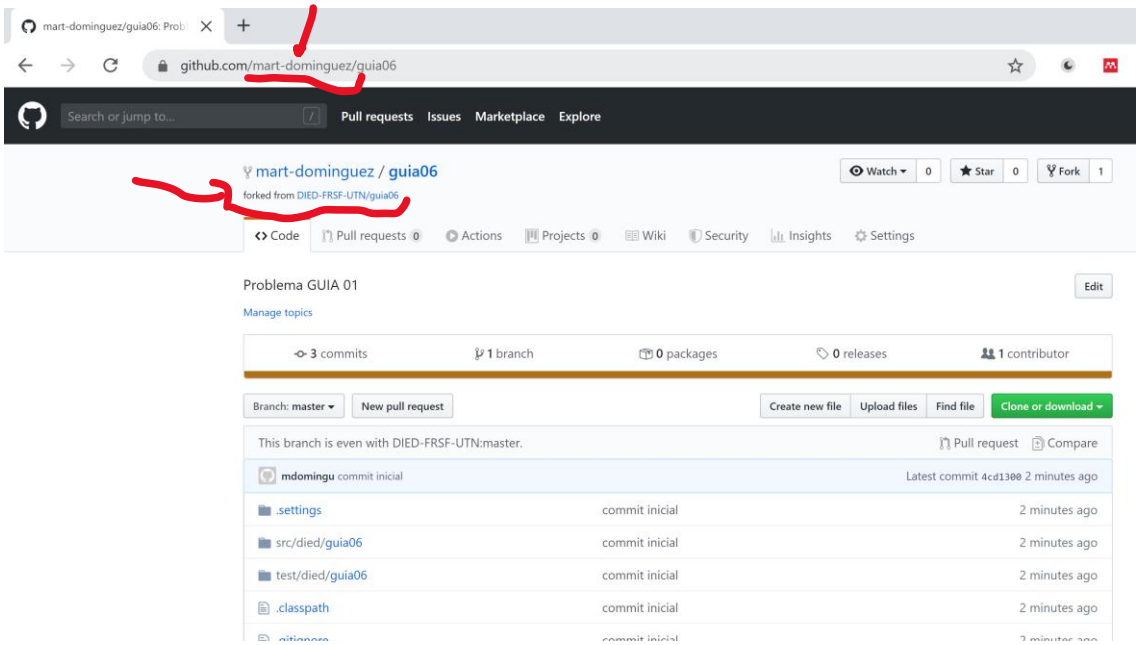
URL del repositorio https://github.com/DIED-FRSF-UTN/guia06

En primer lugar realizar el **FORK**





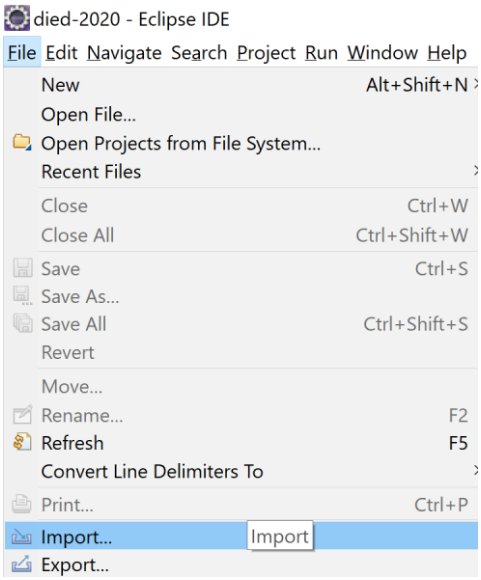
Luego de unos segundos tendrá la copia del repositorio dentro de su usuario de github



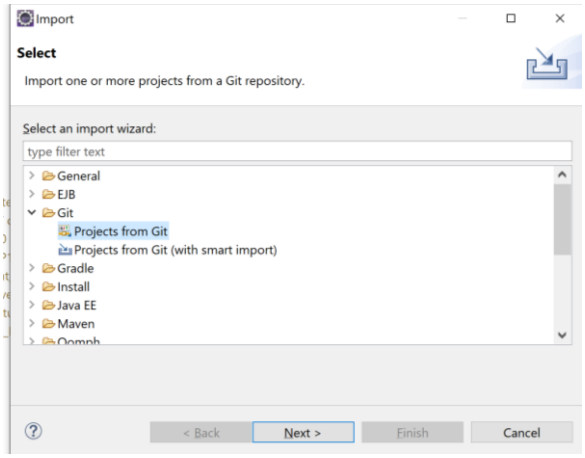
Importar proyecto a Eclipse

Para importar el proyecto git a eclipse, esta vez usaremos una funcionalidad visual de eclipse.

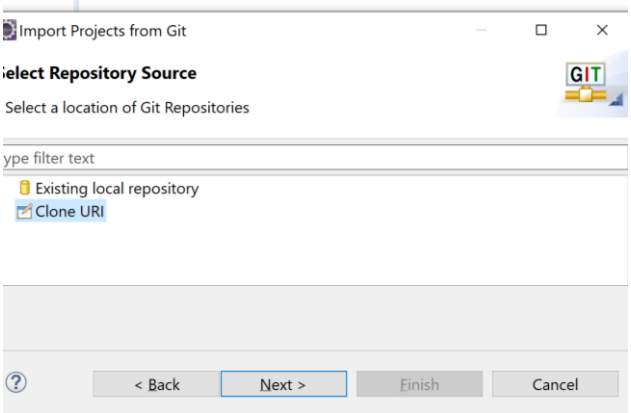
Seleccionar "File"➔ "Import"



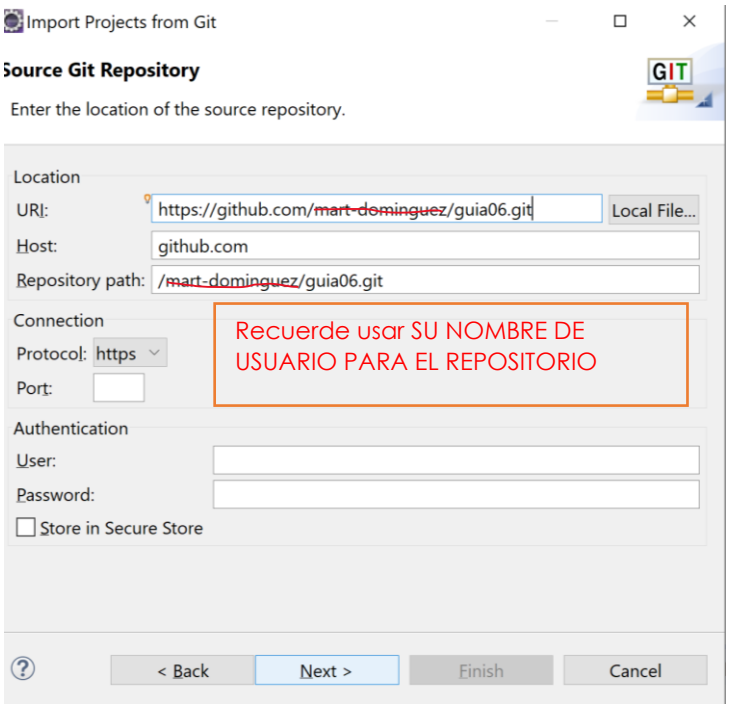
Luego seleccionar "projects from git"



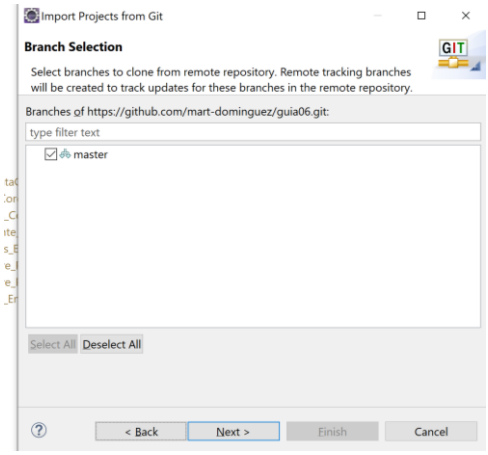
Seleccionar la opción clonar URI

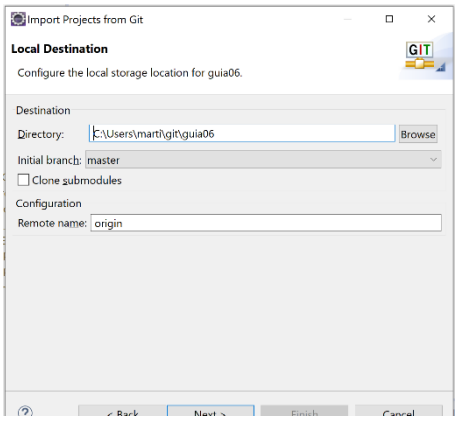


Luego indicar la URL del repositorio GIT de **SU USUARIO**

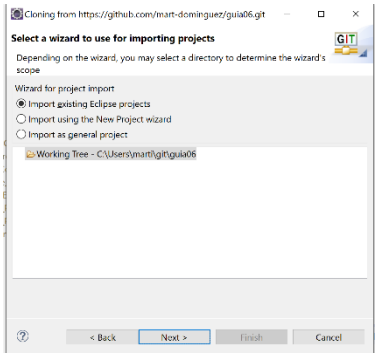


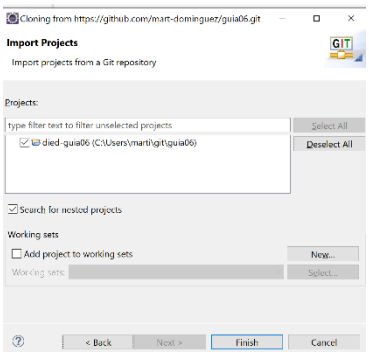
Luego seleccione la rama master (**en este proyecto solo trabajaremos en la rama master) y a continuación** Elegir el directorio local donde guardará el proyecto



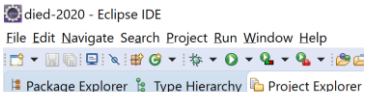


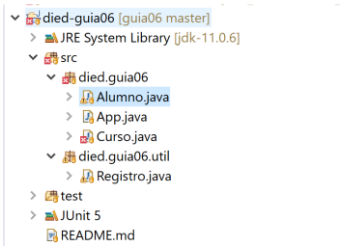
Y finalmente importar el Proyecto a eclipse





Deberá ver el Proyecto en eclipse de la siguiente manera





Descripción del problema

Deberá modelar el caso de una organización que brinda carreras de capacitación a alumnos a través de cursos a los cuales los alumnos se pueden inscribir si cumplen con las situaciones previstas.

De cada Curso se guarda un valor numérico como Identificador (id), el nombre y el ciclo lectivo. Además cada curso conoce la cantidad de créditos que otorga a quienes lo aprueban. Por otra parte cada curso tiene una cantidad de créditos mínimo que el alumno tiene que haber obtenido (aprobando otros cursos) para poder inscribirse. Algunos cursos no requieren créditos mínimos para inscribirse por lo que dicho valor es de cero (0). Además todo curso tiene un cupo, y cuando el mismo es alcanzado no se pueden inscribir más alumnos.

Por otra parte de los métodos que se espera que pueda resolver un curso son:

- Inscribir un alumno: Este método, verifica si el alumno se puede inscribir y si es así lo agrega al curso, y además agrega el curso a la lista de cursos en los que está inscripto el alumno y retorna el valor booleano verdadero. En caso contrario retorna falso y no agrega el alumno a la lista de inscriptos ni el curso a la lista de cursos en los que el alumno está inscripto. Para poder inscribirse un alumno debe
    - a) tener como mínimo los créditos necesarios
    - b) el curso debe tener cupo disponibles
    - c) un alumno puede estar inscripto en simultáneo a no más de 3 cursos del mismo ciclo lectivo.
- Imprimir un curso: se debe poder imprimir el listado de inscriptos ordenados alfabéticamente, por numero de libreta universitaria, o por cantidad de créditos obtenidos.

Además en cada método de la clase curso "registrar" se invoca a una clase que deja registros de auditoria en un archivo del sistema operativo.

A su vez el alumno se modela como una clase que tiene los atributos nombre y nro de libreta universitaria. A su vez cada alumno una vez que le es aceptada la inscripción a un curso, se le agrega dicho curso a su lista de cursos que está

cursando. Por otro lado, cada vez que aprueba un curso, se quita el curso de la lista de cursos que está cursando y se agrega a la lista de cursos aprobados.

El comportamiento **minimo** esperado para la clase alumno es:

- Poder determinar la cantidad de créditos que posee (esto se determina como la suma de los créditos que otorgan los cursos que ya aprobó).
- Poder registrar que se inscribe a un curso.
- Poder registrar que aprobó un curso (y ya no está cursando)

Paso 01 – Capturar excepciones

La clase `died.guia06.util.Registro` posee el método registrar que lanza una excepción, deberá capturarla en los métodos de la clase Curso.

Al finalizar el paso 01 realice la siguiente acción sobre git

```
$ git add .
$ git commit -m “resuelvo paso 01”
$ git push origin master
```

Paso 02 – Implementar los métodos de la clase alumno

Además de los métodos de negocio, debe

- Implementar el método equals, estableciendo que 2 alumnos son iguales si tienen el mismo número de libreta universitaria.
- Implementar la interface comparable, estableciendo que la comparación de 2 alumnos se hace alfabéticamente por el atributo nombre.
- Además, puede crear otros métodos que considere necesarios (getters / setters / constructores).

Al finalizar este paso realice la siguiente acción sobre git

```
- $ git add .
- $ git commit -m “resuelvo paso XX”
- $ git push origin master
```

Paso 03 – Crear test unitarios para los métodos de Alumnos en la clase AlumnoTest

Al finalizar este paso realice la siguiente acción sobre git

```
- $ git add .
- $ git commit -m “resuelvo paso XX”
- $ git push origin master
```

Paso 04 – Implementar los métodos de la clase Curso

Implementar los métodos de negocio de un curso y además puede crear otros métodos que considere necesarios (getters / setters / constructores).

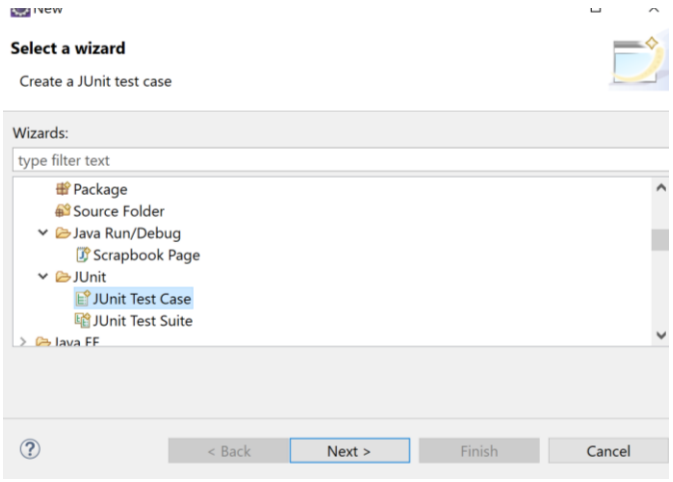
Además para poder implementar el ordenamiento de los alumnos inscriptos por distintos criterios, podría necesitar crear nuevas clases que implementen la interface Compartor<Alumno>

Al finalizar este pas realice la siguiente acción sobre git

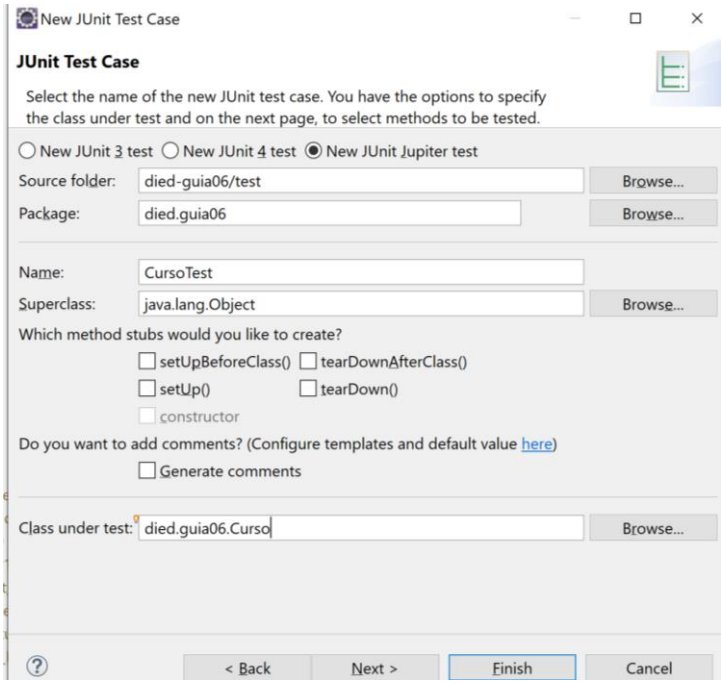
```
-   $ git add .
-   $ git commit -m “resuelvo paso XX”
-   $ git push origin master
```

Paso 05 – Crear una clase de Test para Curso, e implementar los métodos de testing.

- Seleccione: file ➔ new ➔ Junit ➔ Junit test case



Y luego indicar el nombre de la clase de test y el nombre de la clase a testear



Verificar el correcto funcionamiento de las clases de testg

Al finalizar este paso realice la siguiente acción sobre git

```
-   $ git add .
-   $ git commit -m “resuelvo paso XX”
-   $ git push origin master
```

Paso 06: Crear el método main en la clase APP para probar el sistema.

Debe crear los cursos y alumnos necesarios para representar las situaciones descriptas.

Además en caso de ser necesario el método main debe manejar la gestión de errores.

Al finalizar este paso realice la siguiente acción sobre git

```
-   $ git add .
-   $ git commit -m “resuelvo paso XX”
-   $ git push origin master
```

Paso 07: Crear excepciones personalizadas

Agregar a curso el método

- `public void inscribirAlumno(Alumno a) { … }`

Este método ejecutará correctamente si el alumno se pudo inscribir. Caso contrario lanzará una excepción personalizada para cada una de las siguientes situaciones:

- Una excepción si el alumno no tiene los creditos requeridos
- Una excepción que represente que el curso tiene el cupo cubierto
- Una excepción que represente que el alumno ya tiene todas las materias de cursado regular.
- Además este método debe capturar la excepción de entrada salida de la clase registro y en caso de que ocurra relanzarla como una nueva excepción del tipo personalizado “RegistroAuditoriaException”

Al finalizar este pas realice la siguiente acción sobre git

```
-   $ git add .
-   $ git commit -m “resuelvo paso XX”
-   $ git push origin master
```

Paso 08: Agregar métodos de testing para este método

Al finalizar este pas realice la siguiente acción sobre git

```
-   $ git add .
-   $ git commit -m “resuelvo paso XX”
-   $ git push origin master
```

Paso 09: Modificar el método main

Modificar el método main para que use el método para inscribir alumnos que gestiona excepciones, capture las excepciones correspondientes e imprima por consola los mensajes apropiados.

Al finalizar este pas realice la siguiente acción sobre git

```
-   $ git add .
-   $ git commit -m “resuelvo paso XX”
-   $ git push origin master
```