Auxiliar 1

Java Refresher + Github

Profesor: Alexandre Bergel
Auxiliares: Javier Espinoza *(espinozav.javier@gmail.com)*
Eduardo Riveros *(eriveros@dcc.uchile.cl)*
Fecha: 11 de Agosto del 2017

1. Java Refresher

En los archivos adjuntos a la tarea asociada a esta auxiliar (Mini Tarea 0), se les facilitan dos clases ya definidas:

- **CartesianPoint(x,y) :** Punto cartesiano centrado en (x,y). Recibe los siguientes mensajes:
  - *getX()*, devuelve un *double* con la coordenada x del punto
  - *getY()*, devuelve un *double* con la coordenada y del punto
- **AbstractShape(center):**: Figura geométrica, con un *CartesianPoint* equivalente a su centro y posición. Los mensajes que debiese poder recibir una *AbstractShape* son
  - *getArea()*, para obtener un *double* que representa el área de la figura
  - *getPerimeter()*, para obtener un *double* que representa el perímetro de la figura
  - *getPosition()*, para obtener un *CartesianPoint* que representa el centro de la figura
  - *getWidth()*, para obtener un *double* que representa el ancho (en el eje x) de la figura.
  - *getHeight()*, para obttener un *double* que representa la altura (en el eje y) de la figura.
  - *toString()*, para obtener un *String* con el siguiente formato:

    `<Tipo de figura> centrado en (<posición x>, <posición y>)`

  Se les solicita crear nuevas clases que extiendan de *AbstractShape*, para poder representar:

1. Un óvalo y un Rectángulo.
2. Un círculo y un Cuadrado.
3. Cualquier polígono regular con su cara inferior paralela al eje x. *(Especialmente, un triángulo y un pentágono).*

2. Eclipse: ¿Cómo configurarlo?

En la auxiliar notamos que no todos han usado Eclipse antes. Les explicaremos entonces como bajarlo y configurarlo.

2.1. Instalar Eclipse

1. Asegurarse de tener instalado el *Java Development Kit* desde la página de Oracle: **http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html** Recuerden también configurar las Variables de Entorno para poder usar

Java de forma correcta. *(Esto debieron de haberlo hecho para sus tareas de Algoritmos, por lo que omitiremos esa configuración).*

2. Bajar **Eclipse Oxygen** para la plataforma que usan desde la página **https://www.eclipse.org/downloads/**
3. Cuando pregunte por un *Workspace* (Carpeta en que estarán todos sus proyectos), seleccionen una carpeta en que colocarán sus proyectos para el ramo (recomiendo **"Documents/CC3002"**). Pueden seleccionar "*No preguntar la próxima vez"* para que Eclipse siempre use ese *Workspace*.
4. ¡Listo! Ahora puedes importar un proyecto o crear uno desde cero.

2.2. Crear un Proyecto

Para crear un proyecto en Eclipse, sigan estos pasos:

1. Ir a Archivo > Nuevo > Proyecto Java *(File > New > Java Project si está en inglés)*
2. Darle un nombre al proyecto y finalizar.
3. El proyecto aparecerá en "Explorador de Paquetes" (*Package Explorer*), como una carpeta. En el computador, esta carpeta representativa del proyecto se encontrará en la carpeta de su *workspace*.

2.3. Importar un Proyecto

Para importar un proyecto en Eclipse, como el código base de la minitarea de esta auxiliar, seguir estos pasos:

1. Ir a Archivo > Importar *(File > Import*)
2. En la lista que se muestra, ir a General > Proyecto Existente a Espacio de Trabajo (*General > Existing Project from Workspace)*
3. Seleccionar la carpeta correspondiente al proyecto existente. Luego seleccionar los proyectos dentro de esa carpeta que se quieren importar. Ojo que si algún proyecto a importar tiene el mismo nombre que un proyecto que ya está en el *Workspace*, el programa no dejará importar
4. Presionar finalizar.

3. Github: Mi primer Commit

En el Foro Institucional (Perdón por las notificaciones :c) les pedimos que se crearan una Cuenta Educativa en GitHub. ¿Por qué?

- *[De Wikipedia]* Git es un *sistema de control de versiones*, que permite llevar un registro de la historia de cambios de archivos de computadores, y coordinar el trabajo en ellos entre muchas personas. (Piénsenlo burdamente como una versión arcaica de *Google Docs* con historial*)*
- *[También de Wikipedia]* GitHub es un servicio que permite llevar el *control de versiones* de un código fuente de forma colaborativa (como Git), a través del hospedaje de éste en sus servidores. Entre algunas de sus características extras al Git silvestre se encuentran el

control de *issues* y *Bug Tracking*, solicitud de características, administración de tareas y wikis.

Git (y GitHub) son herramientas que usarán mucho en lo que les queda de carrera (en especial en ramos de programar en equipo como *Ingeniería de Software*), por lo que creemos muy buena idea motivarlos a aprenderlas desde ya. Además, nos sirve para llevar registro de su forma de trabajo y motivarlos a que ésta sea periódica (y no lo hagan todo ~~la última semana~~ el último día).

Como Minitarea tendrán que subir el ejercicio realizado en esta auxiliar a un repositorio privado de GitHub. Para hacerlo, deben instalar lo siguiente:

3.1. Pasos Preliminares En Ubuntu/Linux Mint

1. Primero, instalar el paquete con Aptitude.

```
sudo apt-get install git
```

¡Listo!

3.2. Pasos Preliminares En Windows

1. Partir descargando *Git para Windows* de esta URL: https://git-for-windows.github.io/
2. Instalar el fichero anterior, con las opciones predeterminadas en lo posible.

3.3. How to Git Gud

Los siguientes pasos sirven para todas las arquitecturas.

1. Habiendo abierto una ventana de comandos, escribir lo siguiente para llenar la info básica de Git:

```
git config --global user.name "<Mi Nombre>"
git config --global user.email "<mi@correo.cl>"
```

2. Dirigirse a https://www.github.com e iniciar sesión con una cuenta de usuario.
3. En la barra lateral de la página principal *"Your Repositories"*, hacer click en "*New Repository*".
4. Colocar como *repository name* el nombre "minitarea-0" y marcar repositorio como privado. Luego, presionar *Create Repository*.
5. Copiar a la carpeta en que está o estará el proyecto el archivo *.gitignore* que está en material docente. Este archivo sirve para que *Git* no suba archivos compilados o temporales de tu instalación al repositorio central en sus servidores.
6. Tal como señala la página en *GitHub*, abrir un terminal con git en la raíz de donde está o estará el proyecto y ejecutar estos comandos:

```
echo "# minitarea-0" >> README.md
git init
git add README.md .gitignore
git commit -m "primer commit"
git remote add origin https://github.com/<Nombre Usuario Github>/minitarea-0.git
git push -u origin master
```

7. Explicaremos los comandos en orden de aparición:
    - El primer comando crea un *Readme* en la carpeta actual.
    - El segundo inicia Git en el directorio donde se está trabajando.
    - El tercer comando *agrega al paquete a enviar* el archivo *README.md* recién creado
    - El cuarto comando *cierra el empaque de* todos los cambios realizados con el mensaje *first commit*.
    - El quinto comando agrega GitHub como lugar donde mandar los paquetes de código terminado
    - El último comentario envía el paquete, el cual hace que el código se vea actualizado desde la página de Github.

Ahora, cada vez que hagas un cambio dentro de tus archivos de código, debes seguir este mantra en la línea de comandos para *guardar tu progreso* en GitHub:

`git add -A` para agregar todos los archivos cambiados hasta ahora

`git commit -m “<Mensaje con cambios actuales>”` para etiquetar los cambios en el historial de git.

`git push origin master` para enviar los cambios a master en GitHub

Git tiene muchas más funcionalidades entretenidas (como *fork/checkout, merge,* entre otras), pero las mencionadas en esta auxiliar son las esenciales para este semestre.