Primera Práctica
Lenguajes de Programación Orientada a Objetos: Java y Ruby

Segunda sesión

1) Primera parte: Java

A) Descarga el pdf que contiene las cartas[1] de monstruos.

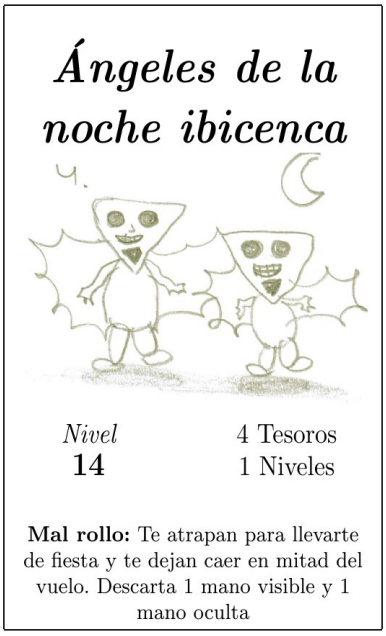
B) En la clase *PruebaNapakalak*i y dentro del método *main()* define una lista de monstruos haciendo uso de la clase ArrayList igual que en la sesión anterior:

```
ArrayList<Monster> monstruos = new ArrayList();
```

C) Construye todos los monstruos que están en el pdf descargado e inclúyelos en la lista definida, para ello primero tienes que ver qué tipo de monstruo tenemos y en base a ello construye un objeto *BadConsequence* (mal rollo) y otro *Prize* (buen rollo). Por ejemplo, para la carta ejemplo de la primera sesión sería:

```
badConsequences = new BadConsequence("Pierdes 5 niveles", 5, 0, 0);
prizes = new Prize(4,2);
monstruos.add("El rey de rosa", 13, badConsequences, prizes);
```

Otro ejemplo, para la siguiente carta, sería:




```
BadConsequences badConsequences = new BadConsequence("Te atrapan
para llevarte de fiesta y te dejan caer en mitad del vuelo. Descarta
1 mano visible y 1 mano oculta",0,
new ArrayList(Arrays.asList(TreasureKind.oneHand)),
new ArrayList(Arrays.asList(TreasureKind.oneHand)));
prizes = new Prize(4,1);
monstruos.add("Ángeles de la noche ibicenca", 14, badConsequences,
prizes);
```

1 Imágenes de las cartas por cortesía de María y Clara.

D) Desarrolla la siguiente consultas, mostrar todos los monstruos (nombre, nivel de combate, buen rollo y mal rollo) que:

- Tienen un nivel de combate superior a 10.
- Tengan un mal rollo que implique sólo pérdida de niveles
- Su buen rollo indique una ganancia de niveles superior a 1
- Cuyo mal rollo suponga la pérdida de un determinado tipo de tesoros ya sea visibles y/o ocultos.

2) Segunda Parte: Ruby

A) Prepara el entorno para trabajar con Ruby

1. Si trabajas en el aula, entra en linux 14.04, abre netbeans y elige la opción de menú Tools/Plugins. En la ventana que se abre, seleccionar la pestaña Downloaded y hacer click en el botón Add Plugins. El plugin (varios archivos) está en la siguiente ruta */usr/local/netbeans-8.0.1/archive/build/updates/* Elegir todos los archivos de la carpeta (salvo `org-netbeans-libs-jellytools-ruby.nbm`) y hacer click sobre el botón Install. Finalmente, reiniciar Netbeans.

   NOTA: Esta operación la tendrás que hacer siempre que entres en el sistema para trabajar con Ruby.

2. Si trabajas con ordenador propio, descarga el plugin para poder trabajar con Ruby en Netbeans en (http://plugins.netbeans.org/plugin/38549). Abre netbeans y elige la opción de menú Tools/Plugins. En la ventana que se abre, selecciona la pestaña Downloaded y pincha en el botón Add Plugins, elige **todos** los archivos anteriormente descargados y pincha sobre el botón Install.

B) Crea el proyecto. Procede de la misma forma que cuando creaste el proyecto Java.

En los siguientes enlaces puedes consultar dudas referentes al lenguaje Ruby:

http://rubytutorial.wikidot.com/ruby-15-minutos, o
http://rubylearning.com/satishtalim/tutorial.html

Y por supuesto, la documentación oficial del lenguaje, bastante buena por cierto

http://www.ruby-doc.org/core-2.1.3/_lib/racc/rdoc/grammar_en_rdoc.html

C) Crea la clase *Prize*, para ello procede igual que se hizo con Java.

1. Define el método *initialize(treasures* y *level)* encargado de la inicialización cuando se construye un objeto *Prize*.
2. *Define los consultores de las variables de instancia, investiga las dos formas de hacerlo.*

D) Define el Modulo *TreasureKind* con nombre las variables globales iguales a los tipos de tesoros e inicializadas a esos valores como String, por ejemplo: ONEHAND = “ONEHAND” y así para todos, la forma de acceder a ellas sería TrearureKind::ONEHAND.

E) Define la clase *BadStuff* con los mismo atributos, inicializadores y consultores que se indicaron en el apartado E de la primera sesión en Java.

F) Define la clase *Monster* con los mismo atributos, inicializadores y consultores que se indicaron en el apartado F de la primera sesión en Java.

G) En la clase PruebaNapakalaki prueba las clases definidas, creando y consultado objetos.