Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC1103 – Introducción a la Programación
2 - 2017

Enunciado Tarea 3

Recordatorio:

- **Fecha de entrega:** 16 de Noviembre de 2017, a las 23:50 hrs.
- **Foro de consulta:** `https://piazza.com/uc.cl/summer2017/iic1103`
- Este trabajo es **estrictamente personal**. Recuerda leer la Política de Integridad Académica del DCC disponible en `http://www.ing.uc.cl/ciencia-de-la-computacion/programas/licenciatura/politica-de-integridad-academica/`. Se usará un software **anti-plagio** para detectar similitud entre códigos.

¡Atención!

Ten en consideración que **no se recibirán entregas fuera del plazo**. Tampoco se evaluará tu tarea si te equivocas al entregar el archivo.
Es tu responsabilidad subir entregas parciales de tu tarea. Se revisará **solamente la última versión** que hayas subido. Revisa la sección **Entrega** para las instrucciones de entrega de tu trabajo.

Objetivo

Aplicar y entender correctamente Programación orientada a objetos, incluyendo los contenidos de archivos, listas y ordenación vistos en clases.

Introducción

Luego de utilizar gran cantidad de tu tiempo para realizar dos duras tareas de programación te das cuenta de que se avecina una fuerte semana de pruebas y no te sobra mucho tiempo para estudiar. Dado esto, decides utilizar lo que has aprendido hasta ahora de programación para simular tu enfrentamiento contra las interrogaciones que tienes por delante, para así estar preparado antes de ser evaluado en tus cursos.

Enunciado

Para esta tarea tendrás que programar una simulación de la preparación y lucha de un alumno contra las pruebas de su semestre universitario.

La simulación consistirá en crear un personaje quien hará el papel del alumno que se enfrenta a las pruebas. El usuario le dará un nombre y le asignará una cierta cantidad de puntos a sus características: vida, destreza, resistencia, suerte e inteligencia. A este conjunto de características le denominaremos *stats*.

La simulación tendrá un total de tres evaluaciones. Al inicio, el usuario podrá equipar a su personaje con un máximo de tres objetos para mejorar ciertas características durante toda la simulación. Además, antes de cada prueba, será posible utilizar consumibles para el mismo propósito, pero éstos surten efecto solamente para ese encuentro.

Una vez que el personaje esté preparado comenzará la evaluación. Dadas las respectivas características del personaje y de la interrogación deberás realizar los cálculos respectivos para determinar cómo le fue al alumno. Estos cálculos se detallan en la sección **Cálculos**. Con cada prueba el personaje irá perdiendo puntos de su característica "vida", a menos que obtenga un 7 ("Perfect"). Si la vida llega a ser menor o igual que cero, quiere decir que el alumno reprobó el curso, por lo tanto, se terminará la simulación [1].

Si el alumno sobrevive al encuentro, entonces tendrá otra instancia para utilizar más consumibles. Una vez preparado, comenzará la siguiente prueba. Este procedimiento se aplica para las tres pruebas de la simulación.

Cuando el jugador haya pasado las tres evaluaciones, entonces el programa debe felicitarlo y preguntar si desea guardar las estadísticas de la partida en un archivo. Luego, la simulación termina.

Clases

Para esta tarea se te exigirá crear y utilizar las siguientes clases:

- Simulacion: Posee los objetos consumibles, equipos, pruebas y al personaje de la simulación.
- Personaje: Posee un nombre y las características del personaje de la simulación.
- Prueba: Posee sus características y el atributo al cual es débil.
- Consumible: Posee un nombre y las características del consumible que representa.
- Equipamiento: Posee un nombre y las características del equipo que representa.

Cabe destacar que esto es lo mínimo que se pide, pero para facilitar tu tarea puedes agregar atributos extra o dejar algunos en otras clases, por ejemplo, dejar el tiempo del personaje en un objeto Partida en vez que en un objeto Personaje. Incluso pueden crear más clases.

Archivos

Antes de comenzar la simulación tu programa debe cargar un archivo llamado `base.txt` en el cual se encuentran los valores de los atributos de los objetos disponibles para la simulación.

[1] Los cursos de la simulación no tienen prueba recuperativa

El archivo `base.txt` tendrá el siguiente formato:

```
VidaBase,Tiempo,PuntosDeStatsInicialesDelJugador
NumeroDisintosDeConsumibles,NumeroDistintoDeEquipamientos
NombreConsumible1,Stock,StatQueAumenta,CuantoAumenta,CostoDeTiempo
NombreConsumible2,Stock,StatQueAumenta,CuantoAumenta,CostoDeTiempo
.
.
.
NombreEquipamiento1,AtributoQueAumenta,Bonificador
NombreEquipamiento2,AtributoQueAumenta,Bonificador
.
.
.
NombrePrueba1,Vida,Destreza,Resistencia,Inteligencia,Suerte,Debilidad
NombrePrueba2,Vida,Destreza,Resistencia,Inteligencia,Suerte,Debilidad
NombrePrueba3,Vida,Destreza,Resistencia,Inteligencia,Suerte,Debilidad
```

La idea es que con estos datos produzcas los objetos para la partida.

Además del archivo `base.txt`, se te entregará un archivo extra llamado `base_extension.txt` en caso que quieras utilizar más objetos. También puedes crear tus propios archivos para probar tu programa, mientras cumplan el mismo formato, pues los ayudantes utilizarán archivos del mismo formato para evaluar.

Al terminar la simulación, el usuario tendrá la opción de guardar los resultados en un archivo de texto con extensión `.txt`. Dicho documento debe guardar las características iniciales y finales del personaje, cuánta vida perdió, con qué accesorios se equipó, cuáles y cuántos consumibles utilizó, y el nombre de las pruebas que realizó.

```
NombreJugador
VidaInicial,DestrezaInicial,ResistenciaInicial,InteligenciaInicial,SuerteInicial,TiempoInicial
VidaFinal,DestrezaFinal,ResistenciaFinal,InteligenciaFinal,SuerteFinal,TiempoFinal
VidaPerdida
NombrePrueba1,NombrePrueba2,NombrePrueba3
Consumible1,CantidadUtilizada
Consumible2,CantidadUtilizada
.
.
.
Equipamiento1
Equipamiento2
Equipamiento3
TiempoTotalInvertido
```

En caso que no tenga equipado tres objetos, entonces el archivo simplemente tendrá menos líneas.

Interacción con el usuario

Luego de cargar el archivo `base.txt`, debes preguntar al usuario por el nombre de su personaje, mostrarle el tiempo, la vida y puntos iniciales que posee. Para que el usuario reparta los puntos entre sus características

debes preguntar por una cantidad entera por atributo. Cabe destacar que no se pueden asignar más puntos de los que se tienen.

```
¡Bienvenido a la simulación de pruebas!
Por favor escribe el nombre de tu personaje:
> Alejandro
Muy bien Alejandro, te informo que posees:
50 puntos de Vida base
1440 puntos de Tiempo
200 puntos para gastar en stats iniciales
¿Cómo quieres repartir tus 200 puntos?
Vida:
>100
Destreza:
>250
Lo siento, esa cantidad excede lo que tienes disponible.
Empezaremos de nuevo por si te equivocaste al repartir.
Vida:
>50
Destreza:
>50
Resistencia:
>40
Inteligencia:
>40
Suerte:
>10
¡Te sobran puntos! Dadas tus malas matemáticas, te las asignaremos a Suerte (puede que la necesites).
Tus stats iniciales son:
Vida: 100 Destreza: 50 Resistencia: 40 Inteligencia: 40 Suerte: 20

Aquí está el listado de todos los equipamientos que hay, elige un número para equiparlo.
En caso de que ya no quieras equiparte más ingresa -1.

1.- Chocolate: Bonificador 1.1 a resistencia
2.- Pañuelos: Bonificador 1.2 a suerte
3.- Botella de agua: Bonificador 1.3 a resistencia
4.- Muñequeras: Bonificador 1.2 a destreza

> 1

1.- Pañuelos: Bonificador 1.2 a suerte
2.- Botella de agua: Bonificador 1.3 a resistencia
3.- Muñequeras: Bonificador 1.2 a destreza

> 2

1.- Pañuelos: Bonificador 1.2 a suerte
2.- Muñequeras: Bonificador 1.2 a destreza

> -1

Aquí están los consumibles, selecciona el número del objeto que deseas.
Para comenzar la evaluación ingresa -1.
Stats actuales: V 100 D 50 R 57 I 40 S 20
Tiempo disponible: 1440
```

```
1.- (1) Ejercicios propuestos: 90 de tiempo, 50 de destreza
2.- (3) Dormir siesta: 30 de tiempo, 40 de resistencia
3.- (2) Usar prenda de la suerte: 10 de tiempo, 10 de suerte

>3
Stats actuales: V 100 D 50 R 57 I 40 S 30
Tiempo disponible: 1430

1.- (1) Ejercicios propuestos: 90 de tiempo, 50 de destreza
2.- (3) Dormir siesta: 30 de tiempo, 40 de resistencia
3.- (1) Usar prenda de la suerte: 10 de tiempo, 10 de suerte

>3
Stats actuales: V 100 D 50 R 57 I 40 S 40
Tiempo disponible: 1420

1.- (1) Ejercicios propuestos: 90 de tiempo, 50 de destreza
2.- (3) Dormir siesta: 30 de tiempo, 40 de resistencia

>-1
```

En caso de sobrar puntos al distribuir los puntos iniciales, estos se asignarán automáticamente al atributo Suerte, tal como se muestra en el ejemplo. En cuanto al equipamiento, al momento de equiparse un objeto, éste debe salir de la lista de equipos disponibles.

Si algún bonificador deja alguna característica con decimales, su valor se redondea hacia abajo, es decir, si era `57.7` queda en `57`.

Para facilitar la representación de los stats en consola se te aconseja implementar el método `__str()__` para algunas clases.

Con respecto a los consumibles, éstos se pueden utilizar hasta que se agoten, siempre y cuando el usuario pueda pagarlos con el atributo tiempo. Si en algún momento el jugador intenta consumir algo que no pueda pagar, el programa debe avisar y no permitir esta acción. Los efectos de los consumibles duran solamente para una prueba. Sin embargo, la cantidad disponible del consumible en la simulación no se restaura, ni tampoco el tiempo del personaje invertido en los consumibles.

Ahora empieza la evaluación. El programa debe presentar a la interrogación contra la cual el jugador se enfrenta, es decir, su nombre y sus atributos. Luego, se muestran los resultados: cuánta vida perdió el personaje. Si el personaje llega a perder todos sus puntos de vida en alguna evaluación, aunque los pierda en la última, la simulación debe terminar súbitamente con un mensaje[2] que dé a entender al usuario que no se cayó el programa, sino que falló en la evaluación.

```
¡Llegó la hora de la evaluación!
Prepárate Alejandro para enfrentar a la Interrogación de Cálculo I (música dramática)
Esta prueba posee V 110 D 30 R 65 I 70 S 30 y es débil contra la destreza ¿Podrás superarla?

2 horas después... y 2 semanas...

¡SÍ! Lo lograste, pero perdiste 16 puntos de vida
```

[2]Una sugerencia podría ser `"Perdiste"`

```
¿Estás listo para el siguiente combate? Mejor prepárate antes.

Aquí est\'an los consumibles, selecciona el número del objeto que deseas.
Para comenzar la evaluación ingresa -1.

Stats actuales: V 64 D 50 R 57 I 40 S 20
Tiempo disponible: 1420
1.- (1) Ejercicios propuestos: 90 de tiempo, 50 de destreza
2.- (3) Dormir siesta: 30 de tiempo, 40 de resistencia

>-1
```

Una vez se termina una interrogación todos los aumentos de los consumibles utilizados desaparecen, pero las cantidades disponibles se mantienen igual. En este momento el programa pregunta, de la misma forma que al inicio, si el personaje desea utilizar más consumibles. No se pregunta por el equipo ya que fue elegido al inicio.[3]

Cuando se terminen todas las evaluaciones y el personaje todavía tenga puntos de vida, entonces el programa lo felicitará, le mostrará sus estadísticas, tal como se muestra en la sección **Archivos** y se le preguntará al usuario si desea guardar los resultados de la simulación en un archivo y qué nombre le pondrá. Finalmente, el programa termina.[4]

Cálculos

Aumento de stats de los consumibles

Simplemente suman una cierta cantidad a la característica que indican. Sin embargo, después de una evaluación este aumento desaparece, es decir, la característica vuelve a su valor antes de utilizar el consumible.

Aumento de *stats* de los equipos

Cada equipo posee un bonificador el cual se multiplica con la característica del personaje que se indica en `base.txt`, luego dicho *stat* toma ese valor mientras el usuario lo tenga equipado. Cabe destacar que los equipos que aumentan la misma característica sí se acumulan, es decir, se multiplica el stat por el bonificador de cada equipo.

Los consumibles se suman aparte de los equipos: primero se calcula el aumento de todos equipos, luego se suman los aumentos de los consumibles.

Evaluación

Al momento de una evaluación, la vida perdida es la suma de las diferencias entre las características del personaje y de la prueba, multiplicado por la división entera de la vida de la prueba por la característica del personaje que sea la debilidad de la prueba. Si este valor resulta ser positivo, entonces se considera que la prueba fue pasada con excelencia y el personaje no pierde vida.

Ejemplo:

[3] No tendría sentido no haberlo equipado desde un principio

[4] Si quieres puedes poner unos créditos al final

Personaje: vida = 100, destreza = 50, resistencia = 57, inteligencia = 40, suerte = 40
Prueba: vida = 110, destreza = 30, resistencia = 65, inteligencia = 70, suerte = 30, debilidad = destreza

cálculo: $((50-30)+(57-65)+(40-70)+(40-30))$ x $(110//50) = (20+0-8-30)$ x $2 = (-18)$ x $2 = -36$

Aquí el personaje pierde 16 puntos de vida.

Bonus

Para esta tarea puedes optar a un bonus que otorga un punto completo a la nota. Sin embargo, debes cumplir con que tu tarea debe superar la nota 4.0 para tener la posibilidad de obtener este bonus.

Backtracking

Luego de terminar tu programa y probarlo algunas veces, te das cuenta que siempre quedas con muy poco tiempo para tu tiempo libre, lo cual siempre es necesario para una vida sana[5]. Dado esto decides optimizar tu simulación de tal forma que te indique qué hacer y qué llevar a tus pruebas para gastar el menor tiempo posible.

Para obtener el Bonus se te pide que tu programa utilice *backtracking* para pasar todas las pruebas con el mínimo gasto de tiempo posible dado un set inicial de *stats*.

Para activarlo, antes de iniciar la simulación debes preguntar al usuario si desea ver la optimización. Si el usuario acepta, entonces debe ingresar los valores iniciales de los stats del personaje, luego el programa solamente debe imprimir un listado de los objetos que utilizó, para cuál evaluación los consumió, con cuánta vida quedó, cuáles accesorios se equipó y cuánto tiempo sobró. No es necesario que se muestre el transcurso de las pruebas. Después el programa termina.

El *backtracking* debe seguir la misma secuencia, es decir, primero debe elegir el equipamiento, luego los consumibles, rendir la evaluación y luego preparase de nuevo para las siguientes pruebas, igual que en la simulación. No basta con optimizar el tiempo para una sola interrogación, sino que debe ser para la simulación completa.

Entrega

Debes guardar tu tarea en un archivo de nombre `tarea3_numero_alumno.py`, donde debes reemplazar `numero_alumno` con tu número de alumno. Por ejemplo, si tu número de alumno es 12345678 el nombre de la tarea sería `tarea3_12345678.py`. **Si no sigues estas instrucciones de entrega, el ayudante corrector tiene el deber de descontar 5 décimas a tu tarea.**

La entrega se realiza a través de un cuestionario en el SIDING disponible en la página web del curso hasta el 16 de Noviembre de 2017 a las 23:50 hrs. **No se recibirán entregas atrasadas ni entregadas por otro medio.**

[5] Y para una salud mental sana también

Indicaciones generales

1. Recuerda que la tarea es **estrictamente individual**. Cualquier situación de copia será sancionada severamente según dicta el código de honor de la universidad, el cual se detalla al final de este documento.

2. **NO hagas la tarea a última hora**, pues el cuestionario se cierra automáticamente cuando se cumple el plazo. **NO se aceptarán entregas atrasadas ni entregadas por otros medios.**

3. Revisa bien lo que entregas, y prueba tu código antes de enviar la versión definitiva. **Puedes enviar tu tarea todas las veces que estimes conveniente, pero solo se almacenará y revisará la última entrega**. Si el ayudante corrector no logra hacer correr tu tarea, es bastante probable que ésta sea evaluada con una nota baja o cercana a 1,0.

Política de Integridad Académica

- Este curso suscribe el Código de Honor, http://www.ing.uc.cl/nuestra-escuela/ingenieria-uc/codigo-de-honor establecido por la Escuela de Ingeniería. En particular, si un estudiante no lo cumple, se le aplicará la Política de Integridad Académica del Departamento de Ciencia de la Computaciń.

Los alumnos de la Escuela de Ingeniería de la Pontificia Universidad Católica de Chile (y alumnos de los cursos dictados por esta Escuela) deben mantener un comportamiento acorde a la Declaración de Principios de la Universidad. En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Académica publicado por la Dirección de Docencia de la Escuela de Ingeniería. Específicamente, para los cursos del Departamento de Ciencia de la Computaciíon, rige obligatoriamente la siguiente política de integridad académica. Todo trabajo presentado por un alumno para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno, sin apoyo en material de terceros. Por "trabajo" se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros. En particular, si un alumno copia un trabajo, o si a un alumno se le prueba que compró o intentó comprar un trabajo, obtendrá nota final 1.1 (uno coma uno) en el curso y se solicitará a la Dirección de Docencia de la Escuela de Ingeniería que no le permita retirar el curso de la carga académica semestral. Por "copia" se entiende incluir en el trabajo presentado como propio partes hechas por otro alumno. En este caso, la sanción anterior se aplicará a todos los alumnos involucrados. Por "compra" se entiende presentar como propio un trabajo hecho por otra persona. Obviamente, está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la referencia correspondiente. Lo anterior se entiende como complemento al Reglamento del Alumno de la Pontificia Universidad Católica de Chile. Por ello, es posible pedir a la Universidad la aplicación de sanciones adicionales especificadas en dicho reglamento.