Texto

Descripción generada automáticamente       

**Instituto Tecnológico y de Estudios Superiores de Monterrey**

**Campus Querétaro**

**Programación orientada a objetos**

**(Gpo 1)**

**Profesor:**

Pedro Oscar Pérez Murueta

**Alumno:**

Oscar Eduardo Nieto Espitia - A01705090

**Actividad:**

Entrega del primer avance del proyecto final

**Programación orientada a objetos**

**Proyecto “Octo-π”**



El profesor Octo-π tiene como objetivo enseñar a todos los niños de primaria, pero hay una complicación, el profesor Octo-π no puede acceder a una computadora ya que todavía no se ha desarrollado un equipo capaz de funcionar el cien por ciento del tiempo debajo de agua salada, así que tienes dos opciones, adaptar una computadora para que Octo-π pueda trabajar o también puedes ayudarlo a automatizar lo que pide el profesor. Como buen estudiante de ingeniería en sistemas ayudas a programar las especificaciones que quiere el Octo-π para que sus alumnos se sientan atraídos al objetivo del profesor.

Ya que el profesor piensa enfocarse en reforzar los conocimientos de todos los niños de primaria decide crear una historia por la que puedas progresar conforme vayas subiendo de nivel enfrentando diversos problemas matemáticos, Octo-π nos narra sobre una rivalidad que ha existido entre él y Shark-τ (Shark-tau), debido a que el piensa que el apellido -τ es más importante que el -π, esto se debe a que en muchos problemas matemáticos es más fácil hacer cálculos con τ (6.28…) que con π (3.14…). Shark-τ afirma que la relación de la circunferencia al diámetro es incorrecta, él dice que los círculos no tienen que ver con los diámetros, sino con radios, por lo que si defines la constante del circulo como la relación de la circunferencia al diámetro, lo que estás haciendo realmente es la relación entre la circunferencia con el doble del radio.

Octo-π

Shark-τ

Por lo tanto, Octo-π quiere recrear las aventuras que vivió con su discípulo enfrentando a la familia τ (tau) y sus hermanos. El jugador tomara el papel del discípulo e intentara resolver los problemas que le lancen los hermanos τ (tau), conforme vayas derrotando a los jefes finales de cada piso (cada respuesta correcta le bajas vida al boss, cada respuesta incorrecta te baja vida a ti) vas a ir avanzando en la historia, ganaras recompensas y experiencia conforme avances de piso.

Cada piso tendrá su pool de preguntas. Todas las preguntas son de opción múltiple y tienen un puntaje, dicho puntaje es la vida que le puedes bajar al boss, el daño lo puedes ir aumentando conforme ganes recompensas y dinero.

Temas según la dificultad:

* Piso 1: Suma y resta de números de dos cifras, multiplicación y división simple, longitud peso y volumen
* Piso 2: Suma y resta de números de cuatro cifras, numeración romana, lectura y escritura de números hasta siete cifras
* Piso 3: Operaciones con fracciones, operaciones con decimales
* Piso 4: Multiplicaciones y divisiones avanzadas, potencias y raíces cuadradas

Como incentivo, el profesor Octo-π desea implementar un sistema de puntos para que el usuario pueda canjear ciertas recompensas, conforme vayas enfrentando y derrotando a los bosses de cada piso vas a conseguir experiencia y Mora (dinero), con el que podrás comprar equipamiento para regenerar vida, la experiencia servirá para mejorar tus stats de vida así como de daño. Al terminar el juego, dependiendo de cuanta Mora (dinero) te haya sobrado, podrás comprar canjes especiales para la materia

|  |  |
| --- | --- |
| **Canje de Puntos** | **Coste de Puntos** |
| Un dulce | 5 |
| Entregar tarea atrasada | 2 |
| Exentar tarea | 5 |
| Exentar proyecto | 30 |
| Exentar examen | 40 |
| Exentar la materia | 100 |

Diagrama

Descripción generada automáticamente

**Explicacion UML**

1. **Clase item**

**Imagen que contiene Diagrama

Descripción generada automáticamente**

Para esta parte del UML están definidos atributos básicos de un item: itemId (servirá como diferenciador de los objetos item, también se agregó este atributo a pesar de que no nos lo pedían en las especificaciones ya que pensé que podría ser más fácil de identificar los ítems por un id y no por un nombre), nombreItem, descripcionItem (Contendrá una pequeña descripción del item así como una pequeña referencia con respecto al lore) y precioItem. A parte de tener los getters y setters también contiene una función toString que juntara todos los atributos de la clase. Decidí implementar herencia en las subclases Poción, Pista y Especial ya que son tipos de ítems, cada uno tendrá su propia pool de ítems y harán diferentes cosas.

|  |  |  |
| --- | --- | --- |
| Poción | Pista | Especial |
| Las pociones modificarán tanto la vida actual del jugador (atributo vida, clase Jugador) como el daño (atributo respuestasCorrectas, clase Jugador). | El item pista tendrá un apuntador al atributo pista de la clase Piso, esto con el objetivo de que todas las preguntas tengan una pista. | El item especial no tendrá ningún uso dentro del juego, pero como Octo-π mencionó, él quería que su juego tuviera un sistema en donde pudieras comprar cosas fuera del juego, por lo tanto esto servirá como canje ante el profesor. |
| Poción - Minecraft Wiki | Hint Svg Png Icon Free Download (#386617) - OnlineWebFonts.COM |  |

1. **Clase Habilidad**

Diagrama

Descripción generada automáticamente

La clase Habilidad es Composición de la clase Jugador a pesar de que la clase Item tiene atributos muy parecidos a la clase habilidad ya que considero que debe de tener otra clase debido a lo que hace referencia, la clase es composición de la clase Jugador ya que la clase habilidad no es dependiente de la clase Jugador, la vida de la clase Habilidad depende de la clase Jugador.

La clase habilidad tiene atributos similares a los de la clase item, por lo tanto, no es necesario describir cada uno de ellos, el atributo en el que se diferencia es powerUp, tiene como fin modificar stats como maxVida que representa las veces que te puedes equivocar al responder a la pregunta, por lo tanto, podrás equivocarte más veces, también modifica las el maxRespuestasCorrectas que representa la cantidad de respuestas correctas que tienes que responder para pasar al siguiente piso, por lo tanto es como si aumentaras tu daño.

Como métodos tiene sus respectivos getters y setters, constructor y una función toString, dicha función se repetirá en casi todas las clases, lo ideal sería definirla como función y así no tener que estar repitiendo código, el problema es que todas las clases tienen diferentes distribuciones de atributos y con mis conocimientos no encuentro la manera de implementarlo a pesar de que usted menciono que se podía hacer.

1. **Clase Jugador**

Texto

Descripción generada automáticamente

La clase Habilidad es composición de la clase Jugador, Un objeto Jugador esta a su vez compuesto por uno o varios objetos de tipo habilidad, el tiempo de vida de los objetos habilidad depende de Jugador, si no existe un jugador no pueden existir sus habilidades, esta fue la lógica que seguí a la hora de asignar composición a esta clase. La clase Item es composición de la clase jugador ya que el jugador puede contener dichos item sin la necesidad de que dichos ítems estén atados a la vida de la clase Jugador.

|  |  |  |
| --- | --- | --- |
| **Atributos clase Jugador** | | |
| **Item** | Es un arreglo de apuntadores que tienen la intención de mostrar todos los atributos del item | Minecraft Potion Chart and Guide - Apex Hosting |
| **Habilidades** | Es un arreglo de apuntadores que tiene la intención de mostrar todos los atributos de la habilidad | Placeholder Image - Skill Icon Png Clipart (#5409408) - PinClipart |
| **Nombre** | Da nombre al jugador, hace más fácil encontrar el objeto | Chunchun |
| **Jugador Id** | Da un id al jugador | 1 |
| **Vida** | Vida actual del Jugador | 5 |
| **Max Vida** | Vida máxima que se puede almacenar | 6 |
| **Experiencia** | Experiencia del jugador, se consigue derrotando al Boss | 23 |
| **Mora** | Moneda del Juego, se consigue derrotando al Boss | 35 |
| **Respuestas Correctas** | Veces que has respondido correctamente a la pregunta del Boss | 3 |
| **Max Respuestas correctas** | Representa el numero de respuestas correctas que debes tener para derrotar el boss, se puede interpretar como tu daño | 6 |

Los métodos de la clase jugador son sus respectivos setters y getters, constructores y el destructor de la clase Jugador, existen funciones especiales como getItems (devuelve el item mediante un apuntador), addItem(agrega un item mediante un apuntador) y removeItem (que remueve el item del jugador), lo mismo pasa para las habilidades.

1. **Clase Piso**

**Tabla

Descripción generada automáticamente**

|  |  |  |
| --- | --- | --- |
| **Atributos clase Piso** | | |
| **Nombre Piso** | Nombre del Piso | El Borde del Abismo / El bosque de la tentación / La Gran Falla / Cáliz de los Gigantes |
| **Numero Piso** | Número del Piso | 1 / 2 / 3 / 4 |
| **Nombre Boss** | Nombre del Boss | Nikikabane / Ottobasu / Mandíbula Escarlata / Shark-τ |
| **Descripción Piso** | Descripción del Piso | --- |
| **Pregunta** | Array de preguntas | [“¿Cuánto es 12 + 49?”, “¿Cuánto es 45 - 32?”, “¿Cuánto es 28 / 4”, “¿Cuánto es 12\*14?”, “¿Carlos tiene 38.5 litros de aceite y ha llenado 8 botellas de tres cuartos de litro cada una. El resto lo pone en frascos de 125 ml cada una. ¿Cuántas frascas llenas?”,] |
| **Respuesta** | Array de respuestas, esta en formato string ya que varias respuestas no son numéricas | [“61”, “13”, “7”, “148”, “268”] |
| **Pista** | En caso de tener duda, existe una pista | [“a”, “a”, “a”, “a”, “Recuerda que un litro equivale a 1000 ml”] |

1. **Clase System**

**Imagen de la pantalla de un celular con texto e imagen

Descripción generada automáticamente con confianza baja**

Las clases Piso y Jugador son agregación de la clase System, necesitan una forma de interactuar entre sí, todavía no poseo el conocimiento de definir sus funciones ni atributos.

**Nota:** Se volvió a plantear todo el UML, por lo tanto toda la sección de la explicación del UML es totalmente nueva al igual que el código. También tomar en cuenta la foto del UML contenido en la carpeta ya que se realizaron cambios muy pequeños (nombres de variables).

Sobre el código: no pude solucionar el error llamado “signal: segmentation fault (core dumped)”, guardar y cargar archivos están como comentarios ya que no logre hacerlos funcionar.