**Proteco - Curso para prebecarios: Java Básico**

**Proyecto: Serie de Ejercicios**

1.- Zoológico (Arreglo de animales, polimorfismo)

Interfaz animal, habrá dos tipos de animales, unos carnívoros (clase padre) como tigres, leones, pumas (clases hijas) y otros herbívoros (clase padre) como antílopes, cebras, venados (clases hijas). Cada uno de los animales tendrá su comida favorita (atributo). Los animales podrán hacer distintas funciones como jugar (mismo tipo, aunque no sea verdad) o comerse a otro animal (tipos distintos) dependiendo el caso (se pasa el animal a interactuar y se checara si es carnívoro o herbívoro). En el main hacer unas pruebas de la interacción entre estos animales, todos los animales serán tratados como “animal” para el pase en parámetros.

2.- Una persona se dirige al PrebeMart a realizar sus compras semanales. Dicha persona no sabe cuántos productos va a comprar y tampoco sabe su valor. Para poder ayudarlo deberás de implementar dos ArrayList. Uno que contenga los nombres de los productos y otro que contenga los precios de dichos productos. Tanto el nombre como el precio, deberán ser ingresados por el usuario, y en seguida, deberán de ser ingresados a las Colecciones. Al momento de pagar le aparece una lista de todos los productos que ha llevado con su respectivo precio, sin embargo, se da cuenta que sólo tiene $500.00, por lo que, si ha excedido su cuenta, deberá elegir unos productos y dejar otros. Para lograr eso, debe poder acceder al índice del producto y así poder elegirlo para eliminarlo de su lista, hasta que finalmente la cuenta sea menor o igual a los $500.00.

El programa debe de preguntar al usuario si desea agregar un producto a su lista, si es así, deberá de ingresar tanto el nombre como el precio a los ArrayList. En caso de que ya no quiera agregar más productos, le aparecerá en pantalla la lista de todos los productos que ha llevado (nombre y precio) así como la suma de los precios. Cuando aparezca la lista, deberá de tener dos opciones:

1) Comprar

2) Dejar productos.

Para la opción “Comprar” deberás de tener validaciones para ver si el dinero que tiene es suficiente. Si el dinero es suficiente, deberá de imprimirse un mensaje que diga “Gracias por su compra!”, en caso contrario debe mandarlo a la opción de “Dejar productos”. Si el usuario elige la opción “Dejar productos” (porque vio en la suma que el dinero no es suficiente) deberá poder ver los índices de los productos en los ArrayList para así poder decidir cuál eliminar. Una vez eliminados los productos, debes imprimir de nuevo la lista de productos a comprar con su precio y darle a elegir de nuevo entre las dos opciones.

3.- Implementa una interfaz con los métodos que tú desees. Crearás 3 clases diferentes que puedan hacer uso de esta interfaz, así como una clase de prueba donde podrás probar su uso. Incluir en comentarios cuál es el objetivo de la interfaz.

4.- Realiza un programa de temática libre en donde puedas utilizar los 4 pilares de la Programación Orientada a Objetos. Intenta utilizar la mayor cantidad de conceptos que se vieron en la primera semana del curso de Java.

5.- Realizar un programa que simplifique una fracción a su mínima expresión. (Si la entrada es de 9/81 por ejemplo, la salida debe ser 1/9).

6.- Una agenda puede guardar una o más personas, y sus funciones son: agregar una persona a la agenda, borrar a una persona de la agenda, buscar a una persona en particular en la agenda, borrar a todas las personas de la agenda.

Una persona tiene un nombre, un teléfono y una mascota.

Una mascota tiene solamente nombre.

Crear un programa que, usando estas clases, permita crear y manipular agendas.

Clases a implementar: Agenda, Persona, Mascota.

7.- Cifrado Cesar Programa que realice un cifrado César. Recibe como argumentos un 1 si se quiere cifrar o un 0 si se quiere descifrar por línea de comandos. Ejemplo: java Cifrado 1 Dependiendo de la bandera se realizará el desplazamiento de 3 caracteres ya sea a la derecha o a la izquierda. Ejemplo:

java Cifrado 1

Cadena ingresada → hola

Cadena cifrada → krñd

java Cifrado 0

Cadena ingresada → dglrv

Cadena descifrada → adiós

8.- Realiza un programa que imprima un rombo en la pantalla del siguiente estilo:



Debe recibir un argumento numérico por línea de comandos que indique el tamaño del rombo a dibujar. Con “1” se dibuja un solo asterisco. Con “2” se dibuja un rombo de 3 filas, con “3” se dibuja uno de 5 filas y así sucesivamente.

9.- Investigar el algoritmo para resolver el juego de las torres de Hanoi para 3 torres y n discos e implementarlo. La salida de este programa debe ser la serie de pasos que se deben seguir para resolver el juego, para cualquier valor de n. Por ejemplo:

Para n=3

1.- mover disco de A a B

2.- mover disco de B a C

…

Etc

Implementarlo orientado a objetos. Tener una clase Torre y una clase Disco.

10.- Elabora un programa que recibe por línea de comandos una serie de números y su salida debe ser todos esos números, expresados en binario, agrupados por el número de 1’s que tienen. Ejemplo: Si la entrada es “2 6 12 3 4 7”, las representaciones binarias de estos números son: “10, 110, 1100, 11, 100, 111” por lo tanto a la salida estos números deben ser mostrados agrupados de la siguiente manera:

10, 100 ← (un 1)

110, 1100, 11 ← (dos 1’s)

111 ← (tres 1’s)

11.- Realizar un programa que calcule el determinante de una matriz nxn.

12.- Pokémon

Pokémon es un juego del estilo RPG lanzado en 1996, ha evolucionado junto con el tiempo hasta convertirse en fenómeno mundial, hoy considerado como un eSport.

Existen miles de copias hechas por los fans conocidas como Hackroms, esta vez, su tarea será diseñar un juego, emulando solo el estilo de RPG.

Para empezar, el jugador podrá escoger un Pokémon inicial, el cual podrá ser entre Totodile (Agua), Cyndaquil (Fuego) y Chikorita (Planta). Este Pokémon será inamovible y deberá estar siempre en el equipo. Cada entrenador tendrá un equipo con máximo 6 Pokémon, ya contando a su inicial. La selección de Pokémon aparte del inicial queda a su elección, se puede consultar toda la pokedex en esta página: http://es.pokemon.wikia.com/wiki/WikiDex

Los Pokémon del equipo deberán tener:

- Vida

- Tipo(s)

- Nivel

- Ataque Físico

- Ataque Especial

- Defensa Física

- Defensa Especial

- Velocidad

Los ataques tendrán:

- Potencia

- Tipo

- Categoría

- Precisión

El daño de estos ataques se rige por una fórmula que ustedes deberán investigar, donde influye tanto nivel del Pokémon, ataque, tipo, etc. Cabe recalcar que este es también un trabajo de investigación, tendrán que checar todos estos datos para hacer su emulador de batallas lo más realista posible.

El flujo de su programa deberá seguir este orden:

1.- Se crean los dos jugadores, uno por uno.

2.- Ambos jugadores escogen a su Pokémon inicial.

3.- Ambos jugadores escogen al resto de su equipo.

4.- Inicia la pelea.

4.1.- Durante la pelea se debe tomar en cuenta todos los aspectos para calcular el daño.

4.2.- El Pokémon en batalla con la velocidad más alta atacará primero.

4.3.- El primer equipo en quedarse sin Pokémon pierde.

5.- Termina la pelea.

5.1.- El perdedor podrá pedir revancha

5.2.- Si el ganador acepta, regresar al punto 1, de lo contrario el juego terminará.

13. Hacer un programa que muestre en pantalla la primera estrofa de "what does the fox say" pero llevado todo a términos de clases y objetos. Usar herencia, interfaces y un arreglo o colección

14. Dadas:

Las clases A, B, C que se encuentren en un paquete llamado “a”

Las clases D, E que se encuentren en un paquete llamado “b”

A debe de contar con un atributo privado, otro público, otro protegido y otro por default, llamados att1, att2, att3, att4 respectivamente.

B y D heredan de A

Tanto A, B, C, D, E son clases públicas y tratan de acceder a att1, att2, att3, att4. Llenar la siguiente tabla con los resultados, escribiendo si ó no si se puede acceder a ese atributo.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Visibilidad | att1 (privado) | att2 (público) | att3 (protegido) | att4 (default) |
| A |  |  |  |  |
| B |  |  |  |  |
| C |  |  |  |  |
| D |  |  |  |  |
| E |  |  |  |  |

\*Cada paquete debe crearse sin usar un IDE

\*La evidencia de este ejercicio son las 5 clases y la tabla en pdf

15. Realizar una “clase perfecta” esta debe abstraer cualquier cosa de la realidad, teniendo como mínimo 2 atributos, y debe sobrescribir cada método de la clase object (a excepción de los “final”), y la sobrescritura debe tener sentido, NO se admite dejar en blanco los métodos o escribir sentencias que no van de acuerdo al objetivo del método dentro del mismo, además investigar e implementar la relación que existe entre los métodos ya que será calificada (por ejemplo la relación entre equals y hashCode()).

En caso de los que son final, escribir comentado su firma y que es lo que hacen.

La clase perfecta debería tener un atributo genérico “<T>”

Un arreglo de objetos de la “clase perfecta” debe ser capaz de ser ordenado (en base a un atributo cualquiera) por el método sort() de la clase Arrays del paquete java.util.

El programa deberá tener una clase de testeo donde se cree un objeto y un arreglo de objetos de la clase perfecta, el cual debe de probar cada método del objeto unitario y debe de ordenar por medio de sort() de Arrays el arreglo de objetos de la clase perfecta.

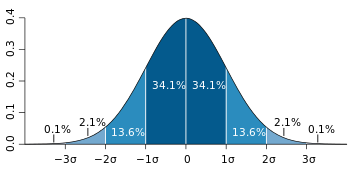
16. Del programa de los animales creado el 4to dia en la clase de java básico, terminarlo y agregar lo siguiente:

* Realizar un método equals a cada reino en base a sus atributos de instancia
* Mejorarlo usando por lo menos un patrón de diseño
* Agregar un atributo de instancia llamado “nombre” a la clase Animal que represente el nombre del animal en cuestión
* Agregar un atributo y método (verídicos) a cada reino y sobrescribir el toString de cada reino.
* Una clase llamada “dios” o “creador” que permita crear un arreglo animales de los 5 reinos y probar todo lo antes mencionado.

17. Realizar una clase que herede de Object, llamada “ClaseInservible” y crear otra clase llamada “ClasePrueba”, que cumpla con lo siguiente:

* Configurar la ClaseInservible de tal modo que no se pueda crear instancias de ClaseInservible desde ClasePrueba
* ClasePrueba no pueda heredar de ClaseInservible
* ClasePrueba puede ejecutar los métodos estáticos de ClaseInservible

18.Simular el tiro de dos dados mediante el uso de la clase Random de Java, el objetivo es comprobar que los resultados de estos n tiros cumplen con una distribución normal.

Se deberá imprimir la gráfica en la consola, la cantidad de tiros será ingresada por el usuario.

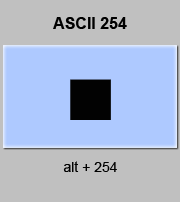
Variables: n tiros

Constantes: caras de dados = 6, dados = 2

19. Dados n números enteros positivos, los cuales representan el nivel de elevación de una pared de ancho 1 en un mapa; calcular cuántos bloques de 1x1 podrán almacenar agua después de una lluvia. Ejemplo:

El siguiente mapa surge del arreglo [0 1 0 2 1 0 1 3 2 1 2 1]

La cantidad de bloques llenos de agua son 6 



* El programa podrá recibir un arreglo (ingresado por el usuario) de cualquier longitud y la altura podrá ser cualquier número entero positivo, se debe imprimir el mapa en la consola (visible al menos para un arreglo de 12 elementos), pueden usar un carácter del código ascii para el cuadrado, no necesariamente el de la imagen.

20. Realizar un gato orientado a objetos con pura terminal.

**Notas:**

* **Cada programa deberá estar correctamente documentado (Investigar que es Javadoc), si no están documentados por javadoc no se calificara (el html es necesario para cada programa).**
* **Todos los programas deberán subirse a Github, en su repositorio creado para el curso de Java, separados en una carpeta utilizada únicamente para este proyecto.**
* **Las personas que no entreguen la tarea de patrón de diseño, se las calificará el proyecto en base al 50%.**
* **Las personas que no tengan derecho a enviar el problema 18 y 19 podrán hacerlo siempre y cuando realicen el ejercicio 18 tomando las constantes como variables por medio de un constructor, tener varios constructores para que asignar valores por default.**
* **La fecha límite de entrega del proyecto será el sábado 6 de octubre a las 11:59 pm.**