Silvio Andrés Orozco Vizquerra
Carné 18282
Fecha de entrega 06 de septiembre, 2018

Ejercicio 7

Parte 1:

1. ¿Qué es una estructura de datos?
   Una estructura de datos es primeramente una colección de datos que se caracterizan por su organización y las operaciones que se definen entre ellos. Esta se puede expresar formalmente, mediante un conjunto de reglas, las relaciones y las operaciones posibles entre ellas. Es decir, es una unión de un conjunto de datos y funciones que lo modifican. Pueden ser representadas como un registro, como un vector o como una simple unión. El objetivo final de una estructura de datos es la forma en que la información será organizada dentro de un programa informático. (Palacios, 2018).
2. ¿Qué dice la Ley de Deméter y para qué sirve?
   La ley de Deméter se expresa como "Habla solo con tus amigos" o ya bien "No aceptes caramelos de extraños". Este concepto se refiere que cada unidad debe tener un limitado conocimientos sobre otras unidades y solo conocer aquellas, que están estrechamente relacionadas a la unidad actual. Solo se puede hablar con amigos inmediatos y nunca con extraños.
   Esto sirve para que un método M de una clase O, solo deberían invocarse métodos de los siguientes tipos de objetos:
   - Del propio objeto O.
   - De los parámetros que recibe el propio método M.
   - De cualquier objeto que instancie el propio método M.
   - De cualquier atributo de O.

   Finalmente, el punto es que las clases solo puedan interactuar entre ellas si están estrechamente relacionadas y sino comunicarse únicamente entre clases amigas para obtener los datos necesarios y de esta forma se obtiene un código más sencillo de mantener, se reduce la duplicación de código, parámetros en los métodos y la cantidad por métodos en clases (Rodríguez, 2018).

Fuentes:
Palacios, A. (2018). *Introducción a algoritmos y estructuras de datos.* Recuperado de http://fcqi.tij.uabc.mx/usuarios/palacios/UNIDAD%201%20%20Introduccion%20a%20los%20algoritmos%20y%20Estructuras%20de%20Datos.pdf

Rodríguez, O. (2018). *Principios de Diseño Orientado a Objetos.* Recuperado de http://oldemarrodriguez.com/yahoo_site_admin/assets/docs/P3-Principios_de_Disen%CC%83o.28794335.pdf

Casado, A. (2015). *Ley de Deméter.* Recuperado de https://www.adictosaltrabajo.com/tutoriales/ley-de-demeter/

Parte 2 y 3:
Diagrama 1 Errores:
Principio de responsabilidad única:
El tablero tiene la responsabilidad de generar las cartas y sacar las cartas, cuando esa debiese ser una función de la baraja en sí. Por tanto, falta una clase de baraja de cartas, que puede sacar una carta.

Principio de alta cohesión y bajo acoplamiento:
El tablero se conecta con Peón, pero esta clase únicamente debería tener relación con la clase jugador y la clase jugador comunicarse con el tablero. Sin embargo, se conectan y esto le da un alto acoplamientos y una baja cohesión.
El random se instancia como un objeto de cartas, por lo que no puede ser utilizado fuera del elemento cartas.

Ley de Deméter:
El tablero obtiene datos directamente del peón a través de un tablero.jugador[1].getpeon().getpeon(). Esto quiere decir que se intenta comunicar con extraños y no con amigos.

Diagrama 2 Correcciones:

Principio de responsabilidad única:
Se ha creado una nueva clase que es BarajadeCartas. Esta es la nueva encargada de tener un arraylist de tipo cartas generadas aleatoriamente. Anteriormente, el tablero tenía la responsabilidad de llamar generar las cartas.

Principio de alta cohesión y bajo acoplamiento:
El tablero ya no está conectado con la clase peón sino únicamente con jugador que es una clase hermana con lo cual se logra independencia de clases y se logra menos responsabilidades para el tablero, y se designan las responsabilidades necesarias al jugador sobre el peón, tales como moverlo.
Random es solo una dependencia de cartas, por lo que puede ser utilizado por cualquier otra clase en el programa y no solo por el objeto cartas.

Ley de Deméter:
El tablero obtiene datos del jugador y posteriormente el jugador obtiene datos del peón, cada uno comunicándose de clase en clase a través de parámetros para hacer surgir una respuesta entre clases no hermanas.