“Poker”

Implementar un algoritmo que entregue el ganador de una mano de póker con varios jugadores, usando solo una baraja con las 52 cartas (4 pintas de 13 cartas cada una). Cada jugador recibe 5 cartas de manera aleatoria. La cantidad de ‘N’ jugadores será menor o igual al máximo que se puede tener dado la cantidad de cartas disponibles.

La implementación debe realizarse en lenguaje de programación JAVA versión 8, IDE/editor de texto libre y no se permite utilizar librerías de terceros (programar en java nativo con posibilidad de utilizar java 8 stream api), ***la solución debe ser una aplicación de consola/terminal***. El tiempo disponible para efectuar esta prueba remota es de 24 hrs a contar de la recepción de la misma, montar el ‘ambiente de desarrollo’ no se considera parte del tiempo.

Observación 1: ‘Reglas de Orden’. Se adjunta la jerarquía de las manos de póker. Ojo que como última regla de desempate (caso borde), se usará la pinta en el siguiente orden (de mayor a menor): Picas (s), Corazones (h), Tréboles (c) y Diamantes (d).

Observación 2: ‘Créditos Extra Opcionales’. Algunas posibles variaciones en el ‘config’ del juego (puedes implementar las que más seguro te hagan sentir, por ejemplo 1,3 y omitir 2)

1) Habilitar la opción para que ingresen los “comodines”. Recordando que para una baraja vienen 2 comodines (1 para cada color, negro y rojo). El uso del comodín es reemplazar por cualquier carta que permita la mejor combinación.
2) Habilitar la opción de jugar la otra versión de póker “Texas Holdem”, que básicamente consiste en que se reparten 7 cartas, y el jugador combina sus mejores 5.
3) Resolver el problema desde un enfoque de programación funcional + POO (utilizando java 8 Stream api por ejemplo)

Criterio de evaluación:

1) Originalidad en los patrones/paradigmas utilizados para resolver/análizar el problema.
2) Simpleza de los algoritmos (sé original pero a la vez eficiente, ej: no es una buena idea resolver este problema de forma recursiva).
3) Modelamiento de estructuras de datos.
4) Principios de POO.
5) Nivel de acople y manejo de dependencia entre las clases que generes (opcional, bonus points).
6) El código debe compilar y ejecutar, por lo tanto debes envíar un breve brief de cómo utilizar tu app.
7) Nivel de detalle sobre aspectos claves de un código fuente prolijo.

Nota: Si detectamos que el algortimo implementado ha sido copiado desde algún sitio de internet (stackoverflow, github, blogs, otros) quedarás fuera de la etapa de selección automáticamente.