Carrera Cavalls

INDEX

1. Introducció

El projecte "Cursa de Cavalls" és una simulació que recrea una emocionant cursa de cavalls mitjançant programació en llenguatge Java. Aquesta simulació consisteix en diverses classes, com ara Cavall, Cursa, HorseRace i Programa, que treballen conjuntament per recrear l'emoció d'una cursa real. A través d'aquest document, explorarem detalladament cada classe, mètode i concepte clau que conforma aquesta simulació.

L'objectiu principal d'aquest projecte és proporcionar una plataforma d'aprenentatge i entreteniment, on es pot explorar el funcionament intern d'una simulació de cursa de cavalls i comprendre com els diferents elements interactuen entre si. Amb aquesta introducció, ens preparem per aprofundir-nos en cada aspecte del projecte, des dels detalls de les classes fins als exemples pràctics d'ús.

-

2. Classes

2.1 Cavall

2.1.1 Detalls de la Classe Cavall:

La classe Cavall modela un cavall participant en una cursa. A continuació, es detallen els aspectes específics d'aquesta classe:

2.1.2 Descripció:

La classe Cavall emmagatzema informació sobre un cavall, com el seu nom (nom), velocitat (velocitat), l'ús d'esteroides anabòlics (hasAnabolicSteroids), temps de finalització (completionTime), temps d'interrupció (interruptedTime), temps real de finalització (realTime), estat de selecció (chosen), i distància recorreguda (distanciaRecorreguda).

2.1.3 Camps:

- **nom** (Tipus: String): Nom del cavall.
- **velocitat** (Tipus: int): Velocitat del cavall amb restriccions.
- **hasAnabolicSteroids** (Tipus: boolean): Indica si el cavall fa ús d'esteroides anabòlics.
- **completionTime** (Tipus: long): Emmagatzema el temps de finalització de la cursa.
- **interruptedTime** (Tipus: long): Emmagatzema el temps d'interrupció de la cursa.
- **realTime** (Tipus: String): Emmagatzema el temps real de finalització de la cursa.
- **chosen** (Tipus: boolean): Indica si el cavall ha estat seleccionat per l'usuari.
- **distanciaRecorreguda** (Tipus: double): Emmagatzema la distància recorreguda pel cavall durant la cursa.

2.1.4 Constructors:

Cavall(String nom, int **velocitat**, boolean **hasAnabolicSteroids**): Constructor que inicialitza les propietats del cavall, prenent en compte les restriccions de velocitat.

2.1.5 Mètodes:

- **setNom**(String nom): Estableix el nom del cavall.
- **setVelocitat**(int velocitat): Estableix la velocitat del cavall amb restriccions basades en l'ús d'esteroides.
- **getNom**(): Retorna el nom del cavall.
- **getVelocitat**(): Retorna la velocitat del cavall.
- **obtenirTemps**(long temps): Estableix el temps de finalització de la cursa per al cavall.
- **setInterruptedTime**(long interruptedTime): Estableix el temps d'interrupció de la cursa per al cavall.
- **getCompletionTime**(): Retorna el temps de finalització de la cursa per al cavall.
- **getInterruptedTime**(): Retorna el temps d'interrupció de la cursa per al cavall.
- **obtenirTempsReal**(String temps): Estableix el temps real de finalització de la cursa per al cavall.
- **getRealCompletionTime**(): Retorna el temps real de finalització de la cursa per al cavall.
- **setDistanciaRecorreguda**(double distanciaRecorreguda): Estableix la distància recorreguda pel cavall.
- **getDistanciaRecorreguda**(): Retorna la distància recorreguda pel cavall.
- **setChosen**(boolean chosen): Estableix l'estat de selecció del cavall.
- **isChosen**(): Retorna si el cavall ha estat seleccionat per l'usuari.

2.1.6 Exemple d'ús:

```
Cavall cavall1 = new Cavall("Troter", 55, false);
cavall1.setChosen(true);
cavall1.obtenirTemps(120000);
System.out.println("Nom del cavall: " + cavall1.getNom());
System.out.println("Velocitat del cavall: " + cavall1.getVelocitat() + "
System.out.println("Temps de finalització: " + cavall1.getCompletionTime(
System.out.println("Selecció de l'usuari: " + cavall1.isChosen());
```

2.2 Cursa:

2.2.1 Detalls de la classe Cursa

La classe Cursa gestiona la informació relativa a una cursa de cavalls. A continuació, es proporcionen detalls específics sobre aquesta classe:

2.2.2 Descripció:

La classe Cursa emmagatzema dades sobre una cursa, com el nom de la cursa (nomCursa), la longitud del recorregut (longitud), la quantitat de cavalls participants (quantitatCavalls), i un indicador que determina si la cursa està en curs (raceOnGoing).

2.2.3 Camps:

- **nomCursa** (Tipus: String): Nom de la cursa.
- **longitud** (Tipus: double): Longitud del recorregut de la cursa en kilòmetres.
- **quantitatCavalls** (Tipus: int): Quantitat de cavalls participants en la cursa.
- **raceOnGoing** (Tipus: boolean): Indica si la cursa està en curs.

2.2.4 Constructors:

Cursa(String **nomCursa**, double **longitud**, int **quantitatCavalls**): Constructor que inicialitza les propietats de la cursa, ajustant les restriccions de la quantitat de cavalls.

2.2.5 Mètodes:

- **setLongitud**(double longitud): Estableix la longitud del recorregut de la cursa.
- **setNomCursa**(String nomCursa): Estableix el nom de la cursa.
- **setQuantitatCavalls**(int quantitatCavalls): Estableix la quantitat de cavalls participants amb restriccions.
- **getLongitud**(): Retorna la longitud del recorregut de la cursa.
- **getQuantitatCavalls**(): Retorna la quantitat de cavalls participants en la cursa.
- **getNomCursa**(): Retorna el nom de la cursa.
- **sortByTime**(List<Cavall> cavalls): Ordena la llista de cavalls per temps de finalització.
- **setRaceOnGoing**(boolean raceOnGoing): Estableix l'estat de la cursa (en curs o finalitzada).
- **getRaceOnGoing**(): Retorna si la cursa està en curs.

2.2.6 Exemple d'ús

```
Cursa cursa = new Cursa("Gran Premi", 15.5, 12);
cursa.setQuantitatCavalls(18);
System.out.println("Nom de la cursa: " + cursa.getNomCursa());
System.out.println("Longitud de la cursa: " + cursa.getLongitud()
System.out.println("Quantitat de cavalls: " + cursa.getQuantitatC
```

2.3 HorseRace:

2.3.1 Detalls de la classe HorseRace

La classe HorseRace representa una cursa de cavalls concurrent que s'executa com a fil. A continuació, es detallen els aspectes específics d'aquesta classe:

2.3.2 Descripció:

La classe HorseRace gestiona la simulació de la cursa per a un cavall concret en el context d'una cursa més gran. Controla la distància recorreguda pel cavall, actualitza la barra de progrés i proporciona informació sobre la cursa.

2.3.3 Camps:

- **c** (Tipus: Cursa): Cursa en què el cavall participa.
- **cavall** (Tipus: Cavall): Cavall que competeix en la cursa.
- **cavalls** (Tipus: List<Cavall>): Llista de tots els cavalls participants.
- **iniciCursa** (Tipus: boolean): Indica si la cursa ha començat o no.

2.3.4 Constructors:

HorseRace(Cavall **cavall**, Cursa **c**, List<Cavall> **cavalls**): Constructor que inicialitza els camps amb les dades del cavall, la cursa i la llista de cavalls.

2.3.5 Mètodes:

- **run**(): Mètode principal que simula la cursa del cavall, actualitza la barra de progrés i imprimeix la informació rellevant.
- **printProgressBar**(int completionRate, double currentSpeed): Imprimeix una barra de progrés per al cavall.
- **formatElapsedTime**(long elapsedTime): Formata el temps transcorregut en el format HH:mm:ss.
- **alternarVelocitat**(Cavall cavall): Alterna la velocitat del cavall de manera aleatòria.
- **stopThread**(): Atura la simulació de la cursa per al cavall.

2.3.6 Exemple d'ús

```
Cavall cavall1 = new Cavall("Troter", 55, false);
Cursa cursa = new Cursa("Gran Premi", 15.5, 12);
List<Cavall> cavalls = Arrays.asList(cavall1, ...); // Altres ca
HorseRace horseRace = new HorseRace(cavall1, cursa, cavalls);
horseRace.start();
```

2.4 Programa:

2.4.1 Detalls de la classe Programa

La classe Programa és l'entrada principal del programa i coordina les operacions, com la simulació de la cursa i la gestió dels cavalls. A continuació, es proporcionen detalls específics sobre aquesta classe:

2.4.2 Descripció:

La classe Programa inicia el programa, interacciona amb l'usuari i coordina la simulació de la cursa de cavalls. Controla la creació de cavalls, la configuració de la cursa i la visualització dels resultats.

2.4.3 Mètodes:

- **Programa**(): Mètode principal que inicia l'execució del programa.
- **createThreads**(Cursa c, List<Cavall> cv): Crea una llista de fils (HorseRace) per a cada cavall.
- **startThread**(List<Thread> t): Inicia tots els fils creats.
- **joinThread**(List<Thread> t): Espera que tots els fils finalitzin.
- **displayHorses**(List<Cavall> cavalls): Mostra els noms de tots els cavalls disponibles.
- **chosenHorse**(List<Cavall> cavalls, String userInput): Assigna el rol de cavall seleccionat.
- **controlAntiDoping**(List<Cavall> cavalls, Cursa c): Realitza un control antidoping als cavalls amb anabolitzants.
- **displayRankingFinal**(List<Cavall> cavalls): Mostra el ranking final de la cursa.

2.5 CreacioCavalls

Només té un String amb 100 noms i un mètode per crear cavalls amb noms aleatoris.

3. Exemples reals

3.1 Cavall

```
while (iniciCursa) {
    double ms = cavall.getVelocitat() / 3.6; // Converteix la v

    // Comprova si la cursa està en curs o si la distància ha e
    if (!c.getRaceOnGoing()) {
        distancia_restant = cavall.getDistanciaRecorreguda();
    }

    // Actualitza la distància restant
    distancia_restant = distancia_restant - ms;

    // Assegura que la distància restant no sigui negativa
    if (distancia_restant < 0) {
        distancia_restant = 0;
    }

    // Actualitza la distància recorreguda pel cavall
    cavall.setDistanciaRecorreguda(distancia_restant);
```

```
cavall.setInterruptedTime(elapsedTime);
cavall.obtenirTemps(elapsedTime);
formattedTime = formatElapsedTime(elapsedTime);
cavall.obtenirTempsReal(formattedTime);
```

```
if (cavall.hasAnabolicSteroids) {
    int boostedNumber = random.nextInt( bound: 20) - 3;
    cavall.setVelocitat(cavall.getVelocitat() + boostedNumber);
```

```
public static void chosenHorse(List<Cavall> cavalls, String userInput){
    for(Cavall c : cavalls){
        if(c.getNom().equals(userInput)){
            c.setChosen(true);
            c.setVelocitat(60);
            c.setHasAnabolicSteroids(true);
        }
    }
}
```

3.2 Cursa

```
int quantitat = cursa.getQuantitatCavalls();
```

```
double distancia_inicial = c.getLongitud() * 1000; // Emmagatzema
double distancia_restant = distancia_inicial;

// Bucle principal que simula la cursa
while (iniciCursa) {
    double ms = cavall.getVelocitat() / 3.6; // Converteix la velo

    // Comprova si la cursa està en curs o si la distància ha esta
    if (!c.getRaceOnGoing()) {
        distancia_restant = cavall.getDistanciaRecorreguda();
    }
```

```
for (int i = 0; i < 3; i++) {
    if (cavalls.get(i).hasAnabolicSteroids) {
        int probability = random.nextInt( bound: 100);
        if (probability > 30) {
            System.out.println("Oh no! " + cavalls.get(i)
            cavalls.get(i).obtenirTempsReal(null);
            c.sortByTime(cavalls);
        }
```

3.3 HorseRace

```
// Actualitza la distància recorreguda pel cavall
cavall.setDistanciaRecorreguda(distancia_restant);

try {
    sleep( millis: 1000); // Pausa el fil durant un segon
    // Calcula la taxa d'acompliment i imprimeix la barra de progrés
    int completionRate = (int) (((distancia_inicial - cavall.getDistanciaRecorreguda()) * 1.0 / d
    alternarVelocitat(cavall);

    boolean finishedPrinting = true; // Declare finishedPrinting outside the synchronized block

    synchronized (lock) {
        // Wait until it's the horse's turn to print
        while (!finishedPrinting) {
            lock.wait();
        }


        // Print the progress bars for all horses
        if(print == 0){
            System.out.println("
            System.out.println("                                    #FASTHORSE
            System.out.println("
            for (Cavall cav : cavalls) {
                int newCompletionRate = (int) (((distancia_inicial - cav.getDistanciaRecorreguda(
                System.out.println(printProgressBar(newCompletionRate, cav.getVelocitat(), cav));
            }
```

```
if(print >= cavallsNull(cavalls)){
    print = 0;
}
```

```
        // Verifica si la distància restant és menor o igual a zero i f
        if (distancia_restant <= 0) {
            stopThread();
        }
    }

    long endTime = System.currentTimeMillis(); // Registra el temps fin
    long elapsedTime = endTime - startTime; // Calcula el temps total t

    String formattedTime = "";

    // Comprova si la cursa està en curs o si ha estat interrompuda
    if (c.getRaceOnGoing()) {
        cavall.setInterruptedTime(elapsedTime);
        cavall.obtenirTemps(elapsedTime);
        formattedTime = formatElapsedTime(elapsedTime);
        cavall.obtenirTempsReal(formattedTime);
    } else {
        elapsedTime = elapsedTime + cavall.getInterruptedTime();
        cavall.obtenirTemps(elapsedTime);
        formattedTime = formatElapsedTime(elapsedTime);
        cavall.obtenirTempsReal(formattedTime);
    }
```

3.4 creacioCavalls

```
creacioCavalls cr = new creacioCavalls();
Cursa cursa = new Cursa( nomCursa: "San Pedro", l
List<Cavall> cavalls = cr.creacioCavall(cursa);
```

3.5 Programa

```
displayHorses(cavalls);
userInput = sc.nextLine();
chosenHorse(cavalls, userInput);

// Inicia els fils i espera que finali
startThread(threads);
joinThread(threads);
```

```
// Control antidopatge
controlAntiDoping(cavalls, cursa);
System.out.println("Control finali
displayRankingFinal(cavalls);
```

4. Conclusió

En aquest projecte, s'ha creat un simulador de curses de cavalls en Java amb diverses classes i funcionalitats. Cada classe té la seva responsabilitat i contribueix a la simulació global de la cursa.

La classe Cavall defineix les característiques dels cavalls, com el nom, la velocitat i la distància recorreguda. La classe Cursa s'encarrega de la configuració de la cursa, la gestió dels cavalls i la classificació final. La classe HorseRace representa la simulació de la cursa com a un fil i controla el progrés dels cavalls.

La classe Programa és l'entrada principal del programa, on s'interactua amb l'usuari per configurar i iniciar les curses. També gestiona la visualització dels resultats i altres aspectes del programa.

L'ús d'objectes, herència i fils ofereix una estructura modular i una simulació realista de curses de cavalls. La interacció amb l'usuari permet una experiència més dinàmica.

Aquesta simulació proporciona una base sobre la qual es poden construir més funcionalitats, com ara l'optimització del rendiment, l'afegit de més característiques als cavalls o la millora de la interfície d'usuari.