UFPR – CST em An[alise e Desenvolvimento de Sistemas
DS142 - Linguagem de Programação Orienta a Objetos

Corrida de Cavalo

Prof. Dr. Dieval Guizelini

2018

Introdução

- Faça um programa que leia N e n nomes e a aposta correspondente ao jogador. Cada aposta pode ser de 1 a 10, correspondente ao número do cavalo. Simule uma corrida de cavalos, criando uma classe Cavalo especializada de Threads em que a cada execução é sorteada um valor entre 5 e 12m e na sequência o cavalo (thread) é colocada para dormir (0,5s).
- Quando o cavalo (objeto thread) percorrer 1000m a corrida e a thread encerra, informando ao objeto monitor a chegada.
- No final (após as 10 threads terem encerradas), o objeto da classe monitor deverá apresentar o rank dos cavalos e os vencedores das apostas.

CorridaCavalo: main

```
public static void main(String[] args) {
      java.util.Scanner scan = new java.util.Scanner(System.in);
       boolean done = true;
       while (done) {
          System.out.println("Simulador de Corrida de Cavalos   (versão texto 0.1)");
          System.out.println("-------------------------------------------------\n");
          System.out.print("Quantos jogadores irão participar dessa corrida (de 1 a 10)? ");
          int numApostadores = 0;
          do {
             numApostadores = scan.nextInt();
             scan.nextLine(); // consome a quebra de linha (ENTER)
             if (numApostadores < 1 || numApostadores > 10) {
                System.out.println("Valor de entrada inválido. Digite um número de 1 a 10.\n");
                System.out.print("Quantos jogadores irão participar dessa corrida (de 1 a 10)? ");
             }
          } while (numApostadores < 1 || numApostadores > 10);
```

CorridaCavalo: main

```
// lê as apostas
String[] jogadores = new String[numApostadores];
int[] apostas = new int[numApostadores];
for (int i = 0; i < numApostadores; i++) {
   System.out.printf("Nome do %do jogador: ", i + 1);
   jogadores[i] = scan.nextLine();
   System.out.printf("Aposta do %do jogador (cavalo de 1 a 10): ", i + 1);
   int aposta = 0;
   do {
      aposta = scan.nextInt();
      scan.nextLine(); // consome a quebra de linha (ENTER)
      if (aposta < 1 || aposta > 10) {
         System.out.println("Valor de entrada inválido. Digite um número de 1 a 10.\n");
         System.out.printf("Aposta do %do jogador (cavalo de 1 a 10)? ", i + 1);
      }
   } while (aposta < 1 || aposta > 10);
   apostas[i] = aposta;
}
```

CorridaCavalo:main

```
JuizMonitor juiz = new JuizMonitor();
// cria as threads e coloca elas para executar
juiz.preparaCorrida(10, 50, 50, 1000);
// da a largada
juiz.iniciaCorrida();
// espera o fim da corrida
juiz.aguardaCorridaEncerrar();
// obtém os resultados
int[] rank = juiz.vencedores();
```

CorridaCavalo:main

```
// apresenta o resultado final
for (int i = 0; i < rank.length; i++) {
    System.out.printf("\nO cavalo %d venceu em %do lugar\n", rank[i], i + 1);
   boolean teveApostador = false;
   for (int j = 0; j < numApostadores; j++) {
       if (apostas[j] == rank[i]) {
           System.out.printf("\t%s\n", jogadores[j]);
           teveApostador = true;
       }
    }
    if( !teveApostador ) {
        System.out.println("\tNinguém apostou no vencedor.");
    }
}
```

CorridaCavalo:main

```
 System.out.print("Jogar novamente (S/N)? ");
          String jogarNovamente = "";
          do {
             jogarNovamente = scan.nextLine();
             if (jogarNovamente.equalsIgnoreCase("S") ||
jogarNovamente.equalsIgnoreCase("N")) {
                 break;
             }
             System.out.print("Jogar novamente (S/N)? ");
          } while (true);
          done = jogarNovamente.equalsIgnoreCase("S");
       }
    }
}
```

JuizMonitor

```
public class JuizMonitor {
    private boolean iniciouCorrida = false;
    private int[] rank = new int[10];
    private int rankIndex = 0;
    private CavaloThread[] cavalos = new CavaloThread[10];

    public JuizMonitor(){
    }
```

JuizMonitor

```
public void preparaCorrida(int minDistance, int maxDistance, int tempoLatencia, int
tamPista) {
    for(int i=0 ; i<cavalos.length ; i++ ) {
        cavalos[i] = new CavaloThread(i+1,this);
        cavalos[i].configura(minDistance, maxDistance, tempoLatencia, tamPista);
        cavalos[i].start();
    }
    try {
        Thread.sleep(100);
    } catch (InterruptedException ex) {
        Logger.getLogger(JuizMonitor.class.getName()).log(Level.SEVERE, null, ex);
    }
    System.out.println("Todos prontos...");
}

public synchronized boolean largou() {
    return iniciouCorrida;
}
```

JuizMonitor

```
public void encerrou(int num) {
    synchronized(this) {
        rank[rankIndex] = num;
        rankIndex++;
    }
}

public synchronized void iniciaCorrida() {
    System.out.println("começou");
    this.iniciouCorrida = true;
}
```

JuizMonitor

```
public void aguardaCorridaEncerrar() {
    while( true ) {
        if( rankIndex >= 10 ) {
            break;
        }
        try { Thread.sleep(500);
        } catch (InterruptedException ex) {
            Logger.getLogger(JuizMonitor.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

```
public int[] vencedores() {
    int[] result = new int[3];
    for(int i=0 ; i<result.length ; i++) {
        result[i] = rank[i];
    }
    return result;
}
```

CavaloThread

```
public class CavaloThread extends Thread {
    private final int num;
    private final JuizMonitor monitor;
    private int menorDistancia = 0;
    private int maiorDistancia = 10;
    private int tempoEspera = 100;
    private int distanciaPercurso = 1000;
    public CavaloThread(int num, JuizMonitor juiz) {
        this.num = num;
        this.monitor = juiz;
    }
    public void configura(int min, int max, int tempo, int tamPista) {
        this.menorDistancia = min;
        this.maiorDistancia = max;
        this.tempoEspera = tempo;
        this.distanciaPercurso = tamPista;
    }
```

CavaloThread

```
@Override
  public void run() {
      System.out.printf("Cavalo %d pronto e
aguardando a largada.\n",num);
      while( !monitor.largou() ) {
         try {
            Thread.sleep(10);
         } catch (InterruptedException ex) {
//...
         }
      }
      // percurso
      System.out.printf("Cavalo %d
largou.\n",num);
      java.util.Random rand = new
java.util.Random();
      int percorreu = 0;
      while( percorreu < distanciaPercurso ) {
         int andou = rand.nextInt(
 maiorDistancia-menorDistancia  ) +
menorDistancia;
         try {
            Thread.sleep(tempoEspera);
         } catch (InterruptedException ex) {
            //
         }
         percorreu += andou;
         System.out.printf(
            "O cavalo %d percorreu %dm\n",
            num,percorreu);
      }
      // avisa o juiz
      monitor.encerrou(num);
   }
```