# Implementando en Othello un xogador “automático”

Na versión realizada ata agora, os dous xogadores eran persoas ás que se lles pedía que escribiran as coordenadas onde querían realizar a xogada. Pero…. ¿ como poderíamos facer para xogar “contra o ordenador” ?  ¿ ou incluso que o “ordendador xogara contra sí mesmo”?.

No xogo que temos deseñado dispoñemos da clase **Player**

```
class Player {
   private String name;      // Nome do xogador
   private int color;        // Neste xogo pode ser 1 ou 2

   public Player(String name) {
       this.name=name;
   }

   public String getName() {
        return name;
   }

   public int getColor() {
        return color;
   }

   public void setColor(int c) {
      color=c;
   }

   /**
    * Elixe a posición da xogada. Lanza unha excepción si a posición elexida non é válida
    */
   public Position doMovement() throws Exception {
      int row;
      char column;
      Position p;
      String line;
      java.util.Scanner scn=new java.util.Scanner(System.in);

      // this almacena a referencia ao obxecto actual (un Player). a JVM chama a toString para convertir o Player
      // na súa representación String
      System.out.print("Xogada de "+this+"\n [1..8][A..Z]?: ");
      line=scn.nextLine();
      row=(int)(line.charAt(0)-'0'); // Recuperamos a primeira letra, e a convertimos nun número
      column=line.charAt(1); // Recuperamos a segunda letra
      return new Position(row,column);
   }

   @Override
   public String toString() {
      return name+" (color: "+Board.strPiece(color)+")";
   }
}
```

A clase **Player** en realidade consiste en:

```
class Player {
   public Player(String name);      // Crea un Player de nome name
   public void setColor(int c);     // Asigna unha cor ao Player
   public String getName();         // Devolve o nome do Player
   public int getColor();           // Devolve a cor asignada ao Player
   public Position doMovement() throws Exception;  // O Player ten a capacidade de facer un movemento
   public String toString();        // O Player devolve como desexa ser representado en String
}
```

O único que diferencia a un xogador humano dun xogador “computerizado” e o xeito de facer o movemento (***doMovement***). O xogador humano realiza o movemento introducindo en teclado a posición onde desexa realizar a xogada. Un xogador “computerizado” tería que dispoñer dun algoritmo que examinara o taboleiro e elexira a posición da xogada en base a unha estratexia.

Entre mellor sexa a estratexia implementada polo algoritmo, mellor xogará o ordenador. Incluso poderíamos deseñar varios tipos de xogadores segundo o nivel da estratexia elexida “básico”, “medio”, “avanzado” … etc.

Facendo uso da herdanza e do polimorfismo é moi simple deseñar un xogador “computerizado” sen facer prácticamente cambios na aplicación, simplemente deseñamos unha clase que herede de Player (que é o xogador humano) e sobrepoñemos o método ***doMovement()*** implementando ahí a nosa estratexia.

O único problema, e que do mesmo modo que un xogador humano pode plantexar a súa estratexia mirando na pantalla o estado do taboleiro,  un xogador “computerizado” necesita poder acceder ao estado do xogo para planificar a súa estratexia. Esto pode facerse incorporándolle un método que lle permita acceder ao obxecto ***Game***. Un xogador “computerizado” necesitará un atributo que garde o obxecto **Game** co que está a xogar.

Logo basta que no arranque do xogo, un dos Players (ou os dous) sexa deste novo tipo.