Hemtentamen TDDC77

för NN - 00000000 – 0000

- Utlämnas: 2010-12-15 kl. 08:00
- Inlämnas senast: 2010-12-16 kl. 09:00
- Jourhavande lärare: Johan Jernlås `johan.jernlas@liu.se` (tfn: 073-941 75 75), tillgänglig via mail under kontorstid, samt på sitt kontor den sista timmen

- Tentamen innehåller 4 uppgifter. Totalt kan erhållas 30p.
- Ungefärliga betygsgränser är 3: 15p, 4: 20p, 5: 25p.
- Tentamen skall redovisas skriftligt i digitalt format (ordbehandlingsprogram eller vanlig textfil) med en tydlig rubrik för varje uppgift/delfråga.
- Inlämning sker genom att svaren mailas till johan.jernlas.liu@analys.urkund.se. När inlämningen är korrekt genomförd så får du ett bekräftelsemail tillbaka.
- Kursen eller tentamen får inte diskuteras med någon annan under tentamenstiden.
- Det är tillåtet att hämta information från böcker/internet/andra källor, men du måste själv formulera dina svar. Man får alltså inte kopiera text.
- Tentamen skall vid inlämningen vara försedd med tentandens namn och personnummer, både i filnamnet och i mailets ämnesrad.
- Mailets ämnesrad skall även innehålla texten ”[TDDC77]”.
- Sent inlämnade tentor blir automatiskt underkända.

1 Teori HT1 (5p)

Förklara noggrant följande begrepp. *Om du har gjort brevet-uppgiften så har du redan full pott på denna deluppgift, skriv bara "brevet" som svar på denna fråga för att indikera detta.* (1p/begrepp)

- körningsfel
- flyttal
- metod
- referenstyp
- returvärde

2 Teori HT2 (5p)

Förklara noggrant följande begrepp. (1p/begrepp)

- åsidosättning
- konstant
- klass
- gränssnitt (interface)
- standardkonstruktor (default constructor)

3 Design av klasshierarkier (10p)

Nedan följer ett antal faktapunkter, din uppgift är att bygga motsvarande klasshierarki. Ange så pass mycket av alla klass- och metoddeklarationer att allt specificerat i uppgiften implementeras (7p). Rita även motsvarande klassdiagram (lättviktsdiagram), denna deluppgift kan lösas utan att lösa den föregående (3p).

Lösningen behöver inte vara komplett, körbar kod. Syntaxdetaljer är oviktiga. Ange alla antaganden du gör för att lösa eventuella problem/tvetydigheter.

- Alla människor är antingen Män eller Kvinnor.
- Människor är familjemedlemmar.
- Familjer består av två eller flera familjemedlemmar.
- Familjer kan känna varandra.
- Husdjur är familjemedlemmar.
- Familjer har semesterplaner.
- En skidsemester är en semesterplan.
- En bilsemester är en semesterplan.
- En solsemester är en semesterplan.

4 Praktiskt kodskrivande (10p)

Komplettera den givna koden enligt anvisningarna i kommentarerna märkta "TODO: ..." (7p). Koden skall skrivas enligt gängse kodkonventioner (2p) och kommenteras (1p).

```
public class Othello {

   public enum Marker{
      EMPTY(" "),
      WHITE("X"),
      BLACK("0");

      private String name;
      Marker(String s){
         name = s;
      }
      public String toString(){
         return name;
      }
   }

   private static final int BOARD_WIDTH = 10;
   private Marker[][] board = new Marker[BOARD_WIDTH][BOARD_WIDTH];
   private Marker currentPlayer = Marker.WHITE;

   public Othello() {
      for (int row = 0; row < BOARD_WIDTH; row++) {
         for (int col = 0; col < BOARD_WIDTH; col++) {
            board[row][col] = Marker.EMPTY;
         }
      }
      /*
       * TODO: Make sure to add the starting markers in the middle
       * like this (O = white, X=black, -=empty):
       * ----
       * -OX-
       * -XO-
       * ----
       **/

      /*
       * TODO: Create a simple game-loop:
       * Print current player, ask for move, do move,
       * (ask again if invalid), print board.
       */

   }

   public void printBoard(){
```

```
        for (int row = 0; row < BOARD_WIDTH; row++) {
            for (int col = 0; col < BOARD_WIDTH; col++) {
                System.out.print(board[row][col]);
            }
            System.out.println();
        }
    }


    /*
     * TODO: Change putMarker and legalMove so that they throw a
     * checked exception called IllegalMoveException if a player
     * tries to make an illegal move.
     */
    public void putMarker(Marker player, int row, int col){
        if (legalMove(player, row, col)){
            board[row][col] = player;
        }
    }

    private boolean legalMove(Marker player, int row, int col) {
        /*
         * TODO return true iff legal row and col,
         * and if the move will result in flipping one or more
         * of the opponents markers. (ie there are one or more
         * markers belonging to the opponent adjacent to this one
         * in a vertical, horizontal or diagonal row terminated by
         * another marker elonging to the current player).
         * 012345
         * ------ 0
         * -X---- 1
         * --O--- 2
         * ------ 3
         *
         * In this case, calling the method for player X with row = 3
         * and col = 3 would yield true.
         */
        return false;
    }


}
```