Hemtentamen TDDC77

December 9, 2014

- Utlämnas Onsdag: 2014-12-10 kl 8:00.
- Inlämnas senast Torsdag: 2014-12-11 kl 9:00.
- Jourhavande lärare: Ahmed Rezine `ahmed.rezine@liu.se`, tillgänglig via mail under kontorstid.

- Tentamen innehåller 4 uppgifter. Totalt kan erhållas 30p.
- Ungefärliga betygsgränser är 3: 15p, 4: 20p, 5: 25p.
- Tentamen skall redovisas skriftlig i digitalt format (ordbehandlingsprogram eller vanlig textfil) med en tydlig rubrik för varje uppgift/delfråga.
- Inlämning sker genom att svaren mailas till `ahmed.rezine.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. **Misstanke om fusk rapporteras till disciplin-nämnden**.
- 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 (men inte i mer än tre rader per begrepp) och ge ett exempel på de följande begrepp. Du måste svara rätt för att få poängerna även om ditt "brev" har fått godkänd. (1p/begrepp)

- alias variabler
- skuggad variabel
- primitiv datatyp
- do-while-slinga
- void

2 Teori HT2 (5p)

Förklara noggrant (men inte i mer än tre rader per begrepp) och ge ett exempel på de följande begrepp. (1p/begrepp)

- åsidosättning (overriding)
- konstruktör (constructor)
- gränssnit (interface)
- private
- undantag (exception)

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 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 eventuella problem/tvetydigheter.

- en `Circle` är en `Shape`
- en `Circle` har en radie som är en `double`
- en `Rectange` är en `Shape`
- en `Rectangle` har en två sidor som är `double`
- en `Triangle` är en `Shape`

- en `Triangle` har en tre sidor som är `double`
- en `Circle` innehåller en lista av andra `Shape`
- en `Shape` är en `AreaComputable`
- en `Shape` är en `PerimeterComputable`
- en `AreaComputable` har en `double getArea()` metod
- en `PerimeterComputable` har en `double getPerimeter()` metod
- Man alltid ange storleken på `Circle`, `Rectangle` eller `Triangle` när man instantiera en.
- Det ska inte gå att instantiera en objekt som är av typ `Shape` men inte `Circle`, `Rectangle` eller `Triangle`.
- Varje `Circle` ska åsidosätta metoden `public toString()`. Det nya metoden ska returnera en `String` som innehåller att det är en cirkel med en viss radie och som också innehåller resultatet av att anroppa på `toString()` av varje `Shape` som tillhör cirkeln.
- Varje `Rectangle` eller `Triangle` ska åsidosätta metoden `public toString()`. Det nya metoden ska returnera en `String` som innehåller att det är en Rectangle eller en Triangle med dess dimensioner.

4 Prakstiskt kodskrivande (10p)

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

```
import java.util.Random;
class NotThreeInARow {

    private boolean[][] board;
    private final int N;

    public NotThreeInARow(int n){
     /* TODO: write a method that
      * initializes N to twice n and board
      * to an array of N arrays of N boolean each,
      * then call setNewRandomPositioning()
      */
    }

    public void setNewRandomPositioning(){
     /* TODO: write a method that
      * assigns each cell in board with a randomly
      * chosen boolean value
      */
    }

    public String toString(){
     String result = "";
```

```
     for(int i=0; i<size; i++){
         for(int j=0; j<size; j++){
          result += (board[i][j]? "1 " : "0 ");
         }
         result += "\n";
     }
     return result;
    }

    private boolean isBalancedRow(int r){
     /* TODO: write a method that
      * returns true if and only if there are
      * as many cells with value true as there are
      * cells with value false in the row r
      */
    }

    private boolean isBalancedCol(int c){
     /* TODO: write a methods that
      * returns true if and only if there are
      * as many cells with value true as there are
      * cells with value false in the column c
      */
    }

    private boolean isValidRow(int r){
     /* TODO: write a method that
      * returns true if and only if row r contains
      * no subsequence (i.e., a number of consecutive cells of r with the
      * same value) of length equal or larger than three.
      */
    }

    private boolean isValidCol(int c){
     /* TODO: write a method that
      * returns true if and only if column c contains
      * no subsequence (i.e., a number of consecutive cells of column c with the
      * same value) of length equal or larger than three.
      */
    }



    public boolean isValid(){
     /* TODO: write a method that
      * returns true if and only if each row and column
      * in board contains the same number of true and false cells,
      * with no three (or more) consecutive cells with the same value.
      */
    }

    public static void main(String[] args){
     NotThreeInARow nt = new NotThreeInARow(4);
     System.out.println(nt);

     while(!nt.isValid()){
         nt.setNewRandomPositioning();
         System.out.println(nt);
     }
    }
}
```