Hemtentamen
Objektorienterad programmering
TDDC77

2008-01-14–15

Individuell examination.

Tid för utlämning: 2008-01-14 ca. 10.30.

Plats för utlämning: John von Neumann på IDA.

Tid för inlämning: 2008-01-15 senast 12.00.

Plats för inlämning: Jonas Wallgrens postfack på IDA.

* Tentamen innehåller 3 uppgifter. Totalt kan erhållas 30p.
  Ungefärliga betygsgränser är 3: 15p, 4: 20p, 5: 25p.
* Tentamen skall redovisas skriftligt, snyggt utfört (helst mha ordbehandlingsprogram).
* Kursen eller tentamen får inte diskuteras med någon annan under tentamenstiden.
* Tentamen skall vid inlämningen vara försedd med tentandens namn och personnummer.
  Om arken inte är hophäftade ska alla ark vara försedda med namn.

1. (10p)

   Förklara i egna ord nedanstående begrepp. Beskriv situationer där de används och/eller problem de underlättar lösningen av.

   - inkapsling
   - subklass
   - arv
   - överlagring
   - åsidosättande
   - abstrakt klass
   - interface
   - multipelt arv
   - grund klon
   - djup klon
   - undantag

2. (10p)

Olika sorters fordon ska beskrivas mha objektorientering.

- Ange så pass mycket av alla klass- och metoddeklarationer att allt specificerat i uppgiften implementeras. Det behöver inte vara komplett, körbar kod. Syntaxdetaljer är oviktiga.
- Ange alla antaganden du gör för att lösa eventuella problem/tvetydigheter.

Relationer mellan fordonssorter:

- Bilar är Landfordon.
- Cyklar är Landfordon.
- Seglare är Sjöfordon.
- Motorbåtar är Sjöfordon.
- Amfibiefarkoster är Landfordon.
- Amfibiefarkoster är Sjöfordon.
- Amfibiefarkoster är Motorfordon.
- Landfordon är Fordon.
- Sjöfordon är Fordon.
- Motorfordon är Fordon.
- Bilar är Motorfordon.
- Motorbåtar är Fordon.

Egenskaper m.m. hos fordon:

- Varje Fordon har information om fart:
  - Varje Sjöfordon har en maxfart och en aktuell fart (i knop).
  - Varje Landfordon har en maxfart och en aktuell fart (i km/h).
- Varje Motorfordon har en motorstyrka.
- Maxfart och motorstyrka sätts vid skapandet av fordonet och ska kunna ändras (t.ex. vid ombyggnad).
- Aktuell fart ska kunna ändras med metoden `sättFart()`.
- Det finns en (t.ex. lagstadgad) högsta tillåtna fart gemensam för alla Sjöfordon. Den ska (t.ex. vid lagändring) kunna ändras med ett enda anrop till `sättHögstaFart()`. Samma sak för Landfordon.
- Det ska finnas en metod `ärSnabbare()` som tar tre argument: två godtyckliga Fordon och ett tredje argument som avgör om det är maxfart eller aktuell fart som ska jämföras.

3. (10p)

I tic-tac-toe har den ene spelare tre ring-brickor och den andre tre kryss-brickor. Målet är att få tre av sina egna i rad horisontellt, vertikalt eller diagonalt på ett 3×3-bråde. Nedanstående är en del av koden för att hantera spelbrädet.

- Komplettera koden så att metoden `getWinner` gör vad kommentaren säger. Lösningen bör vara något intelligentare än en massa if:ar och fullständig uppräkning av alla enskilda fall.
- Ange vad som skulle behöva ändras, och hur, om en uppräkning (enum) ska användas istf nuvarande Marker-klass.

```
public class Marker{
   private int value;
   private Marker(int v){
      value=v;
   }
   public static Marker EMPTY=new Marker(0);    //Tom ruta
   public static Marker RING=new Marker(1);     //Ring-bricka
   public static Marker CROSS=new Marker(2);    //Kryss-bricka
}

public class TicTacToe {
    private Marker [][] board=new Marker[3][3]; //Brädet...
    private TicTacToe(){
        for(int i=0;i<3;i++){
            for(int j=0;j<3;j++){
                board[i][j]=Marker.EMPTY;       //... är tomt från början
            }
        }
    }

    //Här skulle åtminstone behöva finnas metoder för att lägga och
    //flytta brickor och för att på andra sätt komma åt och hantera
    //brädet, men de ingår inte i tentamen.

    public Marker getWinner()//...
        //Returnerar Marker.RING eller Marker.CROSS
        //om resp. har vunnit. Om ingen har vunnit
        //kastas NoOneHasWon.
}
```