Ordinarie tentamen för ID1004 Objektorienterad programmering, 12 juni 2015

Denna tentamen examinerar 3.5 högskolepoäng av kursen.

Inga hjälpmedel är tillåtna.

Tentamen består av en obligatorisk del och en extra del. För ett godkänt betyg på hela tentamen måste den obligatoriska delen vara godkänd. Detta berättigar till betyg E på tentamen. För högre betyg kan ytterligare uppgifter lösas i den extra delen. Examinator förbehåller sig möjligheten att justera tentamensbetyget med hänsyn till en bedömning av helhetsintrycket.

Den obligatoriska delen ger maximalt 37 poäng. För godkänt krävs 24 poäng eller mer.

Läs noga igenom alla frågorna först och planera tiden.

Om inget annat uttryckligen sägs, så visas och efterfrågas kod och uttryck ur Java version 6 eller senare. Var noga med metoders returtyper.

Obligatorisk del

Fråga 1 (3p)

Deklarera och initiera dessa variabler:

a) En heltalsvariabel kallad `counter` som innehåller värdet minus ett.
```
int counter = -1;
```
b) En flyttalsvariabel vid namn `phase` initierad till två gånger värdet av variabeln `angle`.
```
double phase = 2.0 * angle;
```
c) En teckenvariabel som heter `tkn` och har värdet 'A'.
```
char tkn = 'A';
```

Fråga 2 (6p)

Givet denna variabeldeklaration:

```
double [] v = new double[22];
```

Ange för varje programsats nedan om den är korrekt eller inte. Om den inte är korrekt, motivera varför och skriv en korrekt programsats. Programsatserna relaterar endast till variabeldeklarationen, inte till varandra.

a) `v = new double[100];`
korrekt

b) `int k = v[0];`
inte korrekt, elementtypen i v är double och kan inte automatiskt typkonverteras till en int. `int k = (int) v[0];`

c) `v[0] = v[v.length]; // lägg sista elementet först`
inte korrekt, sista elementent i arrayen v har index v.length-1
`v[0] = v[v.length-1];`

```
d) double [] p = v;
```
korrekt

Fråga 3 (2p)

Skapa en ny array med plats för så många flyttal som anges i variabeln `int nofElements`.

```
new double[nofElements];
```

Fråga 4 (1p)

Vilket sanningsvärde beräknas detta boolska utryck till?

```
!(true || false) && !false
!true && !false
false && !false
false && true
false
```

Fråga 5 (2p)

Givet denna kod:

```
float u = distance / radius;
```

Skriv en `if`-sats som utförs när variabeln `u` är mellan -1 – +1 .

```
if (-1 < u && u < 1)
          /* do something */ ;
```

Fråga 6 (2p)

Skriv en `if-else`-sats som garanterar att variabeln `double w` har ett värde inom området 0 – 100.

```
if (w < 0)
          w = 0;
else if (100 < w)
          w = 100;
```

Fråga 7 (1p)

Vilket tal skrivs ut av denna kod?

```
int i = 99;
while (9 < i) {
  i = i / 2;
}
System.out.println(i);
```

Loopen fortgår så länge 9 är mindre än i. Det betyder att det tal som skrivs ut är värdet på i när loop-villkoret inte längre är sant. Variabel i tar värdena 99, 49, 24, 12, 6. Eftersom 6 är mindre än 9 avbryts loopen där och det är 6 som skrivs ut.

Fråga 8 (3p)

Skriv kod som använder en `for`-loop för att välja ut det minsta talet i arrayen `int [] ar`.

```
int min = 0;
for (int i = 1; i < ar.length; i++)
          if (ar[i] < ar[min])
                    min = i;
```

Fråga 9 (3p)

Betrakta metoden `parseNofTimes` nedan:

```
public static int parseNofTimes (String s){
  return Math.max (0, Integer.parseInt(s));
}
```

a) Vilken returtyp har metoden?

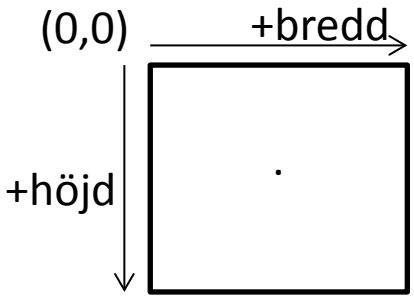
`int`

b) Under vilka förhållanden kan metoden komma att kasta ett undantag (exception)?

Om parametern s har värdet null, eller inte innehåller tecken som kan tolkas som ett heltal.

Fråga 10 (4p)

Skriv en metod `mittPunkt` som beräknar koordinaterna för en rektangels mitt. Parametrarna är bredd och höjd i dubbel precision. Metoden ska returnera en array av heltal där första elementet är x-koordinaten (bredd) och andra elementet är y-koordinaten (höjd). Rektangelns övre vänstra hörn har koordinaterna (0, 0).




```
int [] mittPunkt (double bredd, double höjd) {
  return new int {(int)(bredd/2.0), (int)(höjd/2.0)};
}
```

Fråga 11(10p)

En online-butik säljer böcker, serietidningar, brädspel och film (DVD/Blueray). Modellera i Java de klasser, interface, variabler och metoder som krävs för att avspegla följande krav:

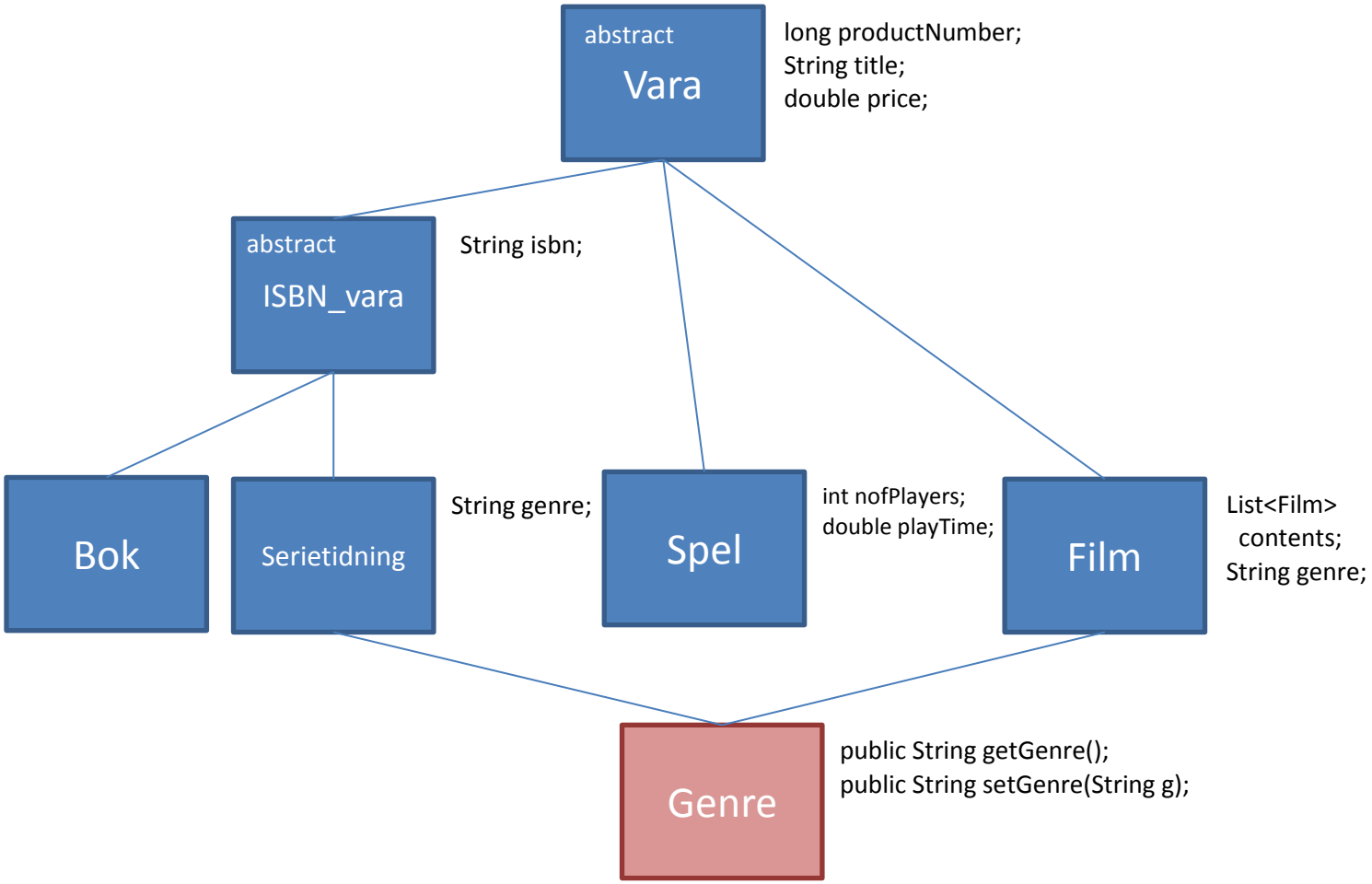
- Alla varor har ett 10-siffrigt produktnummer, en titeltext och ett pris.
- Böcker och serietidningar har en textsträng för ISBN (International Standard Book Number).
- Serietidningar och film har en egenskap *genre* som går är en textsträng. Strängen går att sätta och hämta.
- Brädspel innehåller en uppgift om antal spelare och cirka-tid för det antal timmar det tar att spela ett parti.
- En filmvara kan innehålla andra filmvaror (t ex flera filmer som hör ihop).

- Varje vara ska ha en metod `String toString()` som returnerar en sträng med all information om varan.

Uppgifter:

a) Rita en graf över klasser och gränssnitt där det klart framgår vad som är klasser, arv, interface och implementation av interface. I varje klass och interface ska det tydligt gå att se viktiga variabler och metoder (endast returtyp, metodnamn och parameterlista). Det är tillåtet men inte nödvändigt att skriva Java-kod utanför grafen.

b) Implementera metoden toString() för den klass som modellerar varan serietidning. Om metoden anropar toString() i superklassen så ska även superklassens toString-metod implementeras så att det går att se vad den gör.



```
public abstract class Vara {

  protected long productNumber;
  protected String title;
  protected double price;

  // Konstruktorer och metoder

  public String toString () {
    return String.format("PROD.NR.: %s, TITEL: %s, PRIS: %8.2f",
                              productNumber, title, price);
  }
}

/* ------------------------------------------------ */

public abstract class ISBN_vara extends Vara {
  protected String isbn;

  // Konstruktorer och metoder

  public String toString () {
    return String.format("%s, ISBN: %s", super.toString(), isbn);
  }
}

/* ------------------------------------------------ */

public class Serietidning extends ISBN_vara implements Genre {
  private String genre;

  // Konstruktorer och metoder

  public String getGenre() {
    return genre;
  }

  public String setGenre(String g) {
    genre = g;
  }

  public String toString () {
    return String.format("SERIETIDNING, %s, GENRE: %s",
                          super.toString(), genre);
  }
}
```

Extra del

Med godkänd obligatorisk del, ger poäng på extradelen: 5-6 D, 7-8 C, 9-10 B och 11-13 A.

Fråga 12 (5p)

I Javas standardbibliotek finns klassen `java.awt.Color` som används för att bestämma en färg på skärmen när grafik ritas. En instans av `Color` definierar en viss färg. En av de många konstruktorerna för klassen `Color` ser ut så här:

```
Color(float r, float g, float b)
```

Creates an opaque sRGB color with the specified red, green, and blue values in the range (0.0 - 1.0).

Parametrarna skall alltså vara av typen float, och varje enskilt värde skall vara i intervallet 0.0 – 1.0. Uppgiften är nu att komplettera metoden:

```
Color makeColor (float r, float g, float b) {
  /* Kod saknas här */
}
```

som alltid returnerar en giltig färg även om en eller flera av parametrarna är utanför det tillåtna området. Algoritmen för denna metod är:

Låt min = minimum{r, g, b}
Om min < 0 så
        r += -min, g += -min, b += -min
Låt max = maximum{r, g, b}
Om 1 < max så
        r /= max, g /= max, b /= max
Returnera färgen r, g, b

```
float min = Math.min(Math.min(r, g), Math.min(g, b));
if (min < 0f) {
        r -= min;
        g -= min;
        b -= min;
}
float max = Math.max(Math.max(r, g), Math.max(g, b));
if (1f < max) {
        r /= max;
        g /= max;
        b /= max;
}
return new Color(r, g, b);
```

Fråga 13 (8p)

Conway's Game of Life är en så kallad cellulär automat som består av en (virtuellt) oändlig matris av celler. Varje cell kan vara levande eller död. För att beräkna nästa generation används dessa enkla regler:

- Varje cell omges av exakt 8 grannar.
- En död cell med exakt tre grannar är levande i nästa generation.
- En levande cell med mindre än två eller fler än tre grannar är död i nästa generation.

Antag att en implementation av automaten har följande metod:

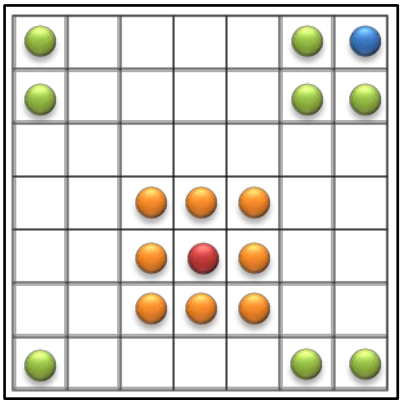
```
void nextGeneration (boolean [][] current, boolean [][] next) {

}
```

Metoden skapar nästa generation genom att läsa från matrisen `current` och skriva till matrisen `next`. Eftersom datorns minne är ändligt, så måste varje implementation begränsa sig till en viss storlek på matrisen. Storleken utläses ur arrayerna med hjälp av `.length`.

Samma ändlighet ställer också till problem i kanterna av matrisen; hörncellerna har bara 3 grannar och kantcellerna mellan hörnen har bara 6 grannar. Lösningen på det är att hitta grannar genom att gå runt kanten på matrisen. I bilden nedan så är den röda cellens granneller orange, och den blå cellens grannar är gröna.

Komplettera metoden `nextGeneration` så att den korrekt skriver nästa generation. En cell som har värdet `false` är död och `true` är levande. Metoden kan tryggt anta att de båda matriserna har samma dimensioner. Metoden måste skriva till varje cell i matrisen `next`. Se upp så att inte en levande cell räknas bland grannarna!



```
public void nextGeneration (boolean [][] current, boolean [][] next) {

  int rows = current.length;    // Antal rader i matrisen
  int cols = current[0].length; // Antal kolumner i matrisen

  for (int r = 0; r < rows; r++) { // För varje rad
    for (int c = 0; c < cols; c++) {  // För varje kolumn i rad r

          // Om denna cell är levande så skall den inte räknas nedan
          int neighbourCount = current[r][c] ? -1 : 0;

          // Grannarnas relativa positioner bestäms med y och x
          for (int y = -1; y < 2; y++) {

            // u är den rad vi får när vi viker runt den övre
            // eller undre kanten på matrisen
            int u = (r + rows + y) % rows;


            for (int x = -1; x < 2; x++) {

              // v är den kolumn vi får när vi viker runt den
              // vänstra eller högra kanten
              int v = (c + cols + x) % cols;

              // Om cellen u,v är levande så addera 1
              // till grannräknaren.
              neighbourCount += current[u][v] ? 1 : 0;
            }
          }

          // Om denna cell (r,c) lever (true)
          // så lever den i nästa generation bara om antalet grannar är
          // två eller tre
          // Annars är cell (r,c) död (false) och blir levande (true) i
          // nästa generation bara om antal grannar är tre.

          if (current[r][c])
            next[r][c] = neighbourCount == 2 || neighbourCount == 3;
          else
            next[r][c] = neighbourCount == 3;

    }
  }
}
```

Bilaga

Följande dokumentation kan vara till hjälp att förstå och lösa vissa uppgifter.

`static double`	**`max`**`(double a, double b)` Returns the greater of two `double` values.
`static float`	**`max`**`(float a, float b)` Returns the greater of two `float` values.
`static int`	**`max`**`(int a, int b)` Returns the greater of two `int` values.
`static long`	**`max`**`(long a, long b)` Returns the greater of two `long` values.
`static int`	**`parseInt(String s)`** **Parses the string argument as a signed decimal integer.**
`static int`	**`parseInt(String s, int radix)`** **Parses the string argument as a signed integer in the radix specified by the second argument.**