Tentamen för TDA540
Objektorienterad Programmering

Institutionen för Datavetenskap
CTH HT-17, TDA540

Dag: 2018-01-13, *Tid:* 14.00-18.00

Ansvarig:	Alex Gerdes
Examinator:	Carlo A. Furia
Förfrågningar:	Alex Gerdes (alexg@chalmers.se, 031-772 6154)
Resultat:	Erhålls via Ladok
Betygsgränser:	3:a 24 poäng 4:a 36 poäng 5:a 48 poäng max 60 poäng
Siffror inom parentes:	Anger maximal poäng på uppgiften
Granskning:	Anslås på kurshemsidan. Vid eventuella åsikter om rättningen ange noggrant vad du anser är fel.
Hjälpmedel:	Cay Horstmann: *Java for everyone* eller Jan Skansholm: *Java direkt med Swing* Understrykningar och smärre förtydligande noteringar får finnas.
Var vänlig och:	Skriv tydligt och disponera papperet på lämpligt sätt. Börja varje uppgift på nytt blad. Skriv ej på baksidan av papperet.
Observera:	Uppgifterna är ej ordnade efter svårighetsgrad. Titta därför igenom hela tentamen innan du börjar skriva. Alla program skall vara väl strukturerade, lättöverskådliga och enkla att förstå. Indentera programkoden! Vid rättning av uppgifter där programkod ingår bedöms principiella fel allvarligare än smärre språkfel.

Lycka till!

Uppgift 1 (18 poäng)

Betrakta klassen `Duplicates` nedan[1]:

```
1   class Duplicates {
2
3      public static void noDuplicates(int[] list) {
4           int[] set = new int[list.length];
5           int size = 0;
6           for (int i = 0; i < list.length; i++) {
7               boolean found = false;
8               for (int j = 0; j < size; j++) {
9                   if (set[j] == list[i]) {
10                      found = true;
11                      break;
12                  }
13              }
14              if (!found) {
15                  set[size] = list[i];
16                  size = size + 1;
17              }
18          }
19          for (int j = 0; j < size; j++)
20              System.out.println(set[j]);
21      }
22
23      public static void main(String[] args) {
24          int[] list1 = {3, 3, 1, 4, 1, 5, 2, 3};
25          noDuplicates(list1);
26      }
27
28  }
```

a) Förklara kort varje *keyword* i klassen. (3 poäng)

b) Lista, i korrekt ordning, alla metodanrop som sker då programmet exekveras. För varje metodanrop ange:

- i vilken klass metoden är deklarerad,
- metodens signatur,
- metodens returtyp, och
- om metoden är statisk eller ej.

(3 poäng)

c) Beskriv programmets 'kontrollflöde' när det exekveras (genom `java Duplicates`) till och med första anropet av `break`. Räkna upp alla exekverade satser (i korrekt ordning) tillsammans med radnummer. Om satsen är ett metodanrop ange också parametrarnas värde. (3 poäng)

d) Vilken utskrift fås då program körs? Förklara utskriften genom en (informell) beskriving av hur `noDuplicates` metoden fungerar. (3 poäng)

[1] Radnumren är inte del av programmet.

e) Ändra programmet:

- Gör så att metoden `noDuplicates`, istället för att skriva ut resultatet, returnerar det. Använd en lämplig returtyp.
- Byta ut innersta `for`-loopen mot en `while`-loop. Bytet skall medföra att `break` satsen försvinner.

(6 poäng)

Uppgift 2

(4 poäng)

Nedanstående kodavsnitt har några (kompilerings) fel. Förklara vad som är fel och rätta till.

```
public static int sum(int[] a) {
  int sum = "0";
  for (int i = 0; i < a.length(); j++)
    sum += a[i];
}
```

Uppgift 3

(8 poäng)

```
class IntPrint {

    public static int strToInt(String s) {
        return Integer.parseInt(s);
    }

    public static void main (String[] args) {
        for (String a: args) {
            try {
                System.out.println(strToInt(a));
            } catch (NumberFormatException e) {
                System.out.println("Fail with " + a);
            }
            finally {
                System.out.println("Moving on");
            }
        }
    }
}
```

a) Vad skrivs ut då programmet körs med följande argument: `12 c 1`? (4 poäng)

b) Vad är den gemensamma superklassen till alla undantag (exceptions)? (1 poäng)

c) Hur ändras programmets beteende om vi ändrar typen i `catch`-blocket från `NumberFormatException` till den gemensamma superklassen till alla undantag? (3 poäng)

Uppgift 4 (16 poäng)

En linjär funktion är en funktion som har följande struktur:

$$y = \text{intercept} + \text{slope} \times x$$

där intercept och slope är konstanter som avgör sambandet mellan x och y. Ovanstående formel kallas för räta linjens ekvation.

- *intercept* kallas eller även konstantterm och bestämmer var linjen skär y-axeln
- *slope* kallas riktningskoefficient och betecknar lutningen på linjen. Om slope är lika med 0 så har linjen en horisontell lutning och linjen ligger därför parallellt med x-axeln. Ett positivt slope ger en linje som lutar snett uppåt åt höger i koordinatsystemet, och ett negativt slope ger en linje som lutar snett neråt åt höger.

Vi har definierat en `BasicLine` klass så här:

```
class BasicLine {

    private float intercept, slope;

    public BasicLine(float intercept, float slope) {
        this.intercept = intercept;
        this.slope = slope;
    }

    float getIntercept() {
        return intercept;
    }

    float getSlope() {
        return slope;
    }
}
```

Din uppgift är att skapa en `Line` klass. `Line` ärver från `BasicLine` och utökar den med följande publika metoder och konstruktorer:

- `Line(float intercept, float slope)` skapar ett objekt med given `intercept` och `slope`.
- `Line(float intercept)` konstruerar en *horisontell* linje med given `intercept`.
- `boolean isHorizontal()` returnerar om linjen är horisontell eller inte.
- `boolean parallel(Line otherLine)` returnerar om linjen är parallellt med `otherLine`.
- `Line clone()` returnerar ett nytt `Line` objekt som är en kopia av den aktuella linjen.
- `float y(float x)` returnerar linjens vertikala koordinat för horisontella koordinat x, dvs vi ska applicera formeln på x.
- `float[] cross(Line otherLine)` returnerar ett fält med två element: första elementet är x-koordinaten och andra är y-koordinaten av punkten där `otherLine` skär aktuella linjen. Om linjerna är parallell med varandra då returnerar metoden `null`.

Uppgift 5 (14 poäng)

Betrakta nedanstående klasser och interfaces:

```
interface A {
    int getX();
}

abstract class B implements A {
    protected int x;
}

interface C {
    void printX();
    void printY();
}

public class D extends B {
    public int getX() {
        return x;
    }
}

public class E extends D implements C {
    public void printX() {
        System.out.println(getX());
    }
    public void printY() {
    }
}

public class F extends E {
    protected int x = 4;
    @Override
    public void printY() {
        super.printX();
    }
}
```

Anta att vi har gjort följande deklarationer:

```
D d = new D();
E e = new E();
F f = new F();
```

a) Hur är `A`, `B`, `C`, `D`, `E` och `F` relaterad till varandra med hänsyn till arv? Rita en *klassdiagram* där alla klasser representeras av en nod och rita en pil mellan nod X och nod Y om Y är den direkta superklassen till X. (4 poäng)

b) Vad är värdet av följande uttryck:

1. `d.getX()`
2. `e.x`
3. `((D)e).getX()`
4. `f.getX()`
5. `f.x`
6. `((D)f).getX()`
7. `((E)f).getX()`
8. `((E)f).x`

(4 poäng)

c) Vad blir utskriften när vi evaluerar följande satser?

1. `e.printX();`
2. `e.printY();`
3. `f.printX();`
4. `f.printY();`

(2 poäng)

d) Bestäm om följande satser är korrekta; om inte förklara varför de inte är korrekta och om det handlar om en kompilerings- eller runtime-fel.

1. `A v1 = new A();`
2. `A v2 = new D();`
3. `C v3 = new D();`
4. `D v4 = new E();`
5. `C v5 = new E();`
6. `F v6 = new D();`
7. `List<D> v7 = new ArrayList<D>();`
8. `ArrayList<D> v8 = new ArrayList<E>();`
9. `A v9 = new B();`

(4 poäng)