Omtentamen för TDA540
Objektorienterad Programmering

Institutionen för Datavetenskap
CTH HT-16, TDA540

Dag: 2017-04-12, *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:	Tentamen kan granskas på studieexpeditionen. Vid eventuella åsikter om rättningen eposta och ange noggrant vad du anser är fel så återkommer vi.
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

(4 poäng)

Vad avses med?

a) Konstruktor

b) Arv

c) Tilldelning

d) Kompilator

Förklara med en eller ett par meningar, du får gärna förtydliga med en skiss eller kod.

Uppgift 2

(4 poäng)

Vilka av raderna 1-22 nedan kompilerar ej? Motiverar varför!

```
1   class Main {
2     public static void main(String[] args) {
3       int n = f(5);
4       double d = f(5.0);
5       d = f(5);
6       d = h(1, 2.0, 3);
7       d = f(1) + h(2, 3);
8       n = b(1.0);
9     }
10
11    int f(int n) {
12      return n + 1;
13    }
14
15    double g(double d) {
16      return d + 1;
17    }
18
19    double h(int n, double d) {
20      return n + d;
21    }
22  }
```

Uppgift 3

(12 poäng)

Skriv en klass `Tokenizer` som delar upp en sträng i substränger (så kallade 'tokens') som är separerade med skiljetecken ('delimiters'). Till exempel, om vi väljer mellanslag och punkt som skiljetecken då har strängen `"Lätt som en plätt."` fyra tokens: `"Lätt"`, `"som"`, `"en"` och `"plätt"`. Klassen `Tokenizer` ska ha följande konstruktor och metoder:

- `public Tokenizer(String input, String delimiters)`
- `public String nextToken()`
- `public boolean hasNextToken()`
- `public List<String> allTokens()`

Metoden `nextToken` returnerar nästa token som finns i strängen. Om det inte finns något 'token' kvar i resten av strängen då kastas ett `NoSuchElementException` undantag. Exempel användning:

```
System.out.println(new Tokenizer("x = x + 1;", " ;").allTokens());
Tokenizer t = new Tokenizer(" Java is the best!!! Duh!", " !");
System.out.println(t.nextToken());
System.out.println(t.allTokens());
System.out.println(t.nextToken());
```

ger:

```
[x, =, x, +, 1]
Java
[is, the, best, Duh]
Exception java.util.NoSuchElementException
```

För att lösa uppgifterna är det tillåtet att använda metoder från `List` gränssnittet och följande metoder från klassen `String`:

- `char charAt(int i)` ger tecknet vid index `i`
- `int indexOf(char ch)` ger index för tecknet `ch` eller -1 om tecknet saknas
- `int length` ger längden av strängen
- `String subString(int start, int end)` ger en delsträng från `start` till `end - 1`
- `String subString(int start)` ger en delsträng från `start` till strängens slut
- `String[] split(String str)` delar upp en sträng i ett fält av delsträngar utifrån ett visst tecken, till exempel:
  `"aaa:bb:cccc:dd".split(":") -> {"aaa", "bb", "cccc", "dd"}`

Uppgift 4

(3 poäng)

Förklara med hjälp av exempel följande begrepp:

a) statisk typ

b) dynamisk typ

c) typkonvertering

Uppgift 5

(5 poäng)

Betrakta metoden `run()` nedan och ange vad skrivs ut när den körs. Hur många variabler finns i programmet (hur många olika behållare för värden kommer att användas)? Hur många objekt är inblandade? Du måste motivera, att bara ange några talvärden ger inga poäng!

```
class Wrapper {
  Integer i;
  Wrapper(Integer i) {
    this.i = i;
  }
}

class Swapper {
  public static void swap(Wrapper x, Wrapper y) {
    Integer tmp = x.i;
    x.i = y.i;
    y.i = tmp;
  }
}

void run() {
  Swapper ws = new Swapper();
  int a = 1;
  int b = 2;
  Wrapper x = new Wrapper(a);
  Wrapper y = new Wrapper(b);
  ws.swap(x, y);
  System.out.println("a = " + x.i + " b = " + y.i);
}
```

Uppgift 6

Ett heltal P är ett primtal om det är större än 1 och inte har några andra positiva delare än 1 och talet självt. Ett heltal som är inte ett primtal kallas för en *komposit*. Till exempel, $2, 3, 7$ är primtal medan $4 = 2 * 2$ och $12 = 2 * 2 * 3$ är komposit.

a) Skriv en metod `isPrime` som kontrollerar om ett heltal `n` är ett primtal och returnerar `true` i sådan fall och annars `false`. Kom ihåg att uttrycket `x % y` resulterar i resten av heltalsdivisionen av `x` med `y`. Lösningen ska ha lämplig typsignatur för metoden `isPrime` samt dess implementation. (3 poäng)

b) Skriv en metod `isComposite` som tar ett heltal `n` och returnerar `true` om `n` är en komposit, och `false` annars. Implementation av `isComposite` ska använda metoden `isPrime`. (1 poäng)

c) Skriv en metod `primes` som tar ett heltal `n` som input och returnerar ett heltalsfält med *alla* primtal från `2` till `n` i numerisk ordning, så att fältets längd är lika med antal primtal mellan `2` och `n`. Återigen använd metoden `isPrime` i implementationen. (6 poäng)

Uppgift 7

Betrakta följande klass- och gränssnittsdeklarationer:

```
interface Vehicle {
  String registrationId();
}

interface Wheeler {
  int numWheels();
}

abstract class FourWheeler implements Vehicle, Wheeler {
  int numWheels() {
    return 4;
  }
}

class Car extends FourWheeler {
  String registrationId = "unknown";
  String registrationId() {
    return registrationId;
  }
}

class Volvo extends Car {
  String manufacturedIn() {
    return "Made in Göteborg";
  }
}
```

a) Vad skriver följande satser ut? (2 poäng)

1. `System.out.println(new Car().numWheels());`
2. `System.out.println(new Volvo().numWheels());`
3. `System.out.println(new Car().registrationId());`
4. `System.out.println(new Volvo().registrationId());`
5. `System.out.println(new Volvo().manufacturedIn());`

b) Förklara för varje nedanstående sats om den blir accepterad eller resulterar i ett kompileringsfel, om det är fel förklara varför. (10 poäng)

1. `Vehicle v = new Vehicle();`
2. `Vehicle v = new Car();`
3. `Car c = new Volvo();`
4. `Volvo c = new Car();`
5. `FourWheeler f = new Volvo();`
6. `FourWheeler f = new FourWheeler();`
7. `Wheeler w = new Car();`
8. `Volvo v = new Volvo();`
9. `List<Car> = new List<Car>();`
10. `List<Vehicle> = new List<Car>();`

Uppgift 8

Betrakta följande deklaration för klassen `IntList` som representerar en heltalslista:

```
class IntList {
  private final Integer[] data;
  private int size;

  public IntList() {
    data =  new Integer[10_000];
    size = 0;
  }

  public int size() {
    return this.size;
  }
}
```

Implementationen använder ett privat fält `data` för att spara alla element i listan. Instansvariabeln `size` räknar antal element i listan, som kan läsas av med metoden `size()`. Elementen i listan är sparade konsekuitivt i fältet `data` från position `0` till och med position `size - 1`. Syftet är att kompletera implementationen av klassen `IntList` med följande metoderna.

a) Skriv en metod `get` med typsignatur

```
public Integer get(int index)
```

som returnerar elementet sparat på position `index` i listan. Om `index` är ogiltigt då returnerar `get` värdet `null`. (2 poäng)

b) Skriv en metod `remove` med typsignatur

```
public Integer remove()
```

som tar bort elementet längst till höger i listan och returnerar detta. Om listan är tom då returnerar `remove` värdet `null`. (4 poäng)

c) Skriv en metod `add` med typsignatur

```
public void add(Integer item)
```

som lägger till ett heltalsvärde `item` längst till höger av element i listan. Om det finns inget utrymme kvar i listan då kastas ett `FullException` undantag. (4 poäng)