LULEÅ TEKNISKA UNIVERSITET

Tentamen i
Objektorienterad programmering och design
Totala antalet uppgifter: 5
Lärare: Håkan Jonsson, 491000, 073-820 1700
Resultatet offentliggörs senast: 2016-09-15.

Tillåtna hjälpmedel: Inga.

Kurskod	D0010E
Datum	2016-08-25
Skrivtid	5 tim

OBS! Lösningar som baseras på annat ur Javas standardbibliotek än `Object` och `RuntimeException` (som fritt får kastas där så är lämpligt) ger inga poäng. Eventuella undantag från detta anges i uppgiftstexten.

---

1 Teori

a) Arv är centralt i objektorienterad programmering. Varför finns arv och vad är fördelarna med arv? Svara med att punkta upp de anledningar och fördelar du känner till. (1,5p)

b) Rita UML-diagram över programdelarna på sid. 4 ("Kod till uppgift 1b"). (1,5p)

c) I uppgift 5 beskrivs bl a hundar, valpar och nyfödda valpar. Om nu metoden `rapportera` nedan anropas med arrayen

```
Hund[] d = {new Hund("A"), new NyföddValp("B"), new Hund("C"),
            new Valp("D"), new Valp("E"), new NyföddValp("F")};
```

vad (om något) skrivs då ut? (1,5p)

```
void rapportera(Hund[] d) {
  if (d.length > 0) {
    double s = 0;
    String str = "För ";
    for (int i = 0; i < d.length; i++) {
      s = s + d[i].skatt();
      str = str + d[i].namn();
      str = str + (i == d.length - 1 ? "" :
                   (i == d.length - 2 ? " och " : ", "));
    }
    System.out.println(str + " blir det " + s + " kr");
  }
}
```

d) Den interna klassen `Node<E>` nedan används för att bygga enkellänkade nodkedjor (listor).

```
class Node<E> {
  E content;
  Node<E> next;
}
```

Skriv en rekursiv metod `int size(Node<E> list)` som returnerar antalet noder i listan som `list` refererar till. Om `list` är `null` är listan vara tom och antalet 0. Annars beror antalet av vad listan som `list.next` refererar till innehåller. (1,5p)

2 Veckodag

Denna uppgift ger maximalt 4p om lösningen använder `if`- eller `switch`-satser. Annars, utan `if`- och `switch`-satser men t ex med indexering in i arrayer, kan den ge maximalt 6p.

Uppgift: Skriv en metod `public static String veckodag(String datum)` som givet ett datum under 2016 i form av en sträng `2016-MM-DD` (t ex 2016-08-25, den 25:e augusti 2016) returnerar vilken veckodag det är (Torsdag).

I din lösning ska du använda metoderna `substring` och `parseInt` som beskrivs på sidan 5. Du behöver inte kontrollera att `datum` är på rätt form. (Max 6p)

—

Om

m = månadens kod enligt vänstra tabellen nedan,
d = siffrorna `DD` tolkade som ett heltal dvs dagen i månaden och
f = resten vid heltalsdivision mellan $6 + m + d$ och 7.

så får man svaret ur den högra tabellen nedan.

Månad	Kod (c)
Januari	-1
Februari	2
Mars	3
April	6
Maj	1
Juni	4
Juli	6
Augusti	2
September	5
Oktober	0
November	3
December	5

Kod (f)	Veckodag
0	Lördag
1	Söndag
2	Måndag
3	Tisdag
4	Onsdag
5	Torsdag
6	Fredag

3 Heltalsmängder

Skriv en klass `BegränsadMängd` för storleksbegränsade mängder av heltal. En mängd är en slags behållare som aldrig innehåller dubletter. Nyskapad är sådana tomma. Klassen ska innehålla följande konstruktor och metoder:

- `public BegränsadMängd(int max)` som skapar en mängd som kan innehålla upp till `max` tal.
- `public void läggTill(int tal)` som gör att `tal` finns i mängden. Det spelar ingen roll om `tal` redan fanns i mängden eller ej. Skulle däremot gränsen `max` överskridas om `tal` las till, så ska undantag kastas (välj själv lämpligt undantag; du behöver inte implementera undantagsklassen) och mängden förbli oförändrad.
- `public void taBort(int tal)` som säkerställer att `tal` inte finns i mängden. Det spelar ingen roll om `tal` fanns i mängden eller ej.
- `public boolean har(int tal)` returnerar `true` om, och endast om, mängden innehåller `tal`.

- `public int storlek()` returnerar aktuellt antal element i mängden.
- `public int max()` returnerar `max`.

Exempel: Sätter vi in $1, 3, 5, 1, 2, 2, 5$ och 4 i en nyskapad mängd med `max`= 100 så hamnar $1, 2, 3, 4$ och 5 i mängden, som har storleken 5. Här skulle `har(1)` ge `true` medan `har(8)` skulle returnera `false`. Anropas nu `taBort` i tur och ordning med argumenten $3, 2, 2, 3, 5$ och 2 så innehåller mängden sen endast elementen 1 och 4, varför `storlek()` skulle returnera 2. (6p)

4 Obegränsade mängder

Någon snäll person har lagt till en metod `public Iterator iterator()` till klassen `BegränsadMängd` i uppgift 3. Denna metod returnerar, som namnet antyder, ett iteratorobjekt som kan användas för att få mängdens alla element ett i taget. Såna iteratorobjekt har två metoder:

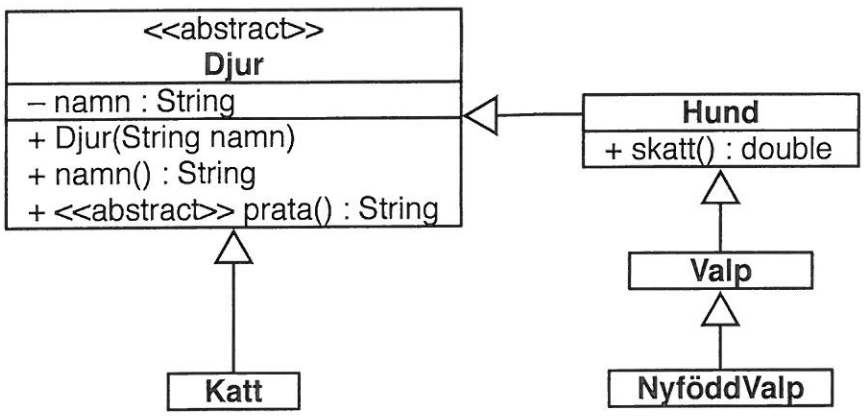
- `public int next()` ger ett ("nästa") heltal ur mängden som inte returnerats vid tidigare anrop till `next`.
- `public boolean hasNext()` ger `true` så länge det finns fler element att få `next` och annars `false`.

Iteratorn påverkar inte mängden. Om `next()` anropas i ett läge då `hasNext()` skulle returnera `false` kastas undantaget `NoSuchElementException`.

Uppgift:

a) Skriv en klass `ObegränsadMängd` för heltalsmängder utan storleksbegränsning. Klassen ska ha samma metoder som `BegränsadMängd` förutom `max()` och iteratormetoderna. Internt ska uppgiften elementen lagras i `BegränsadMängd`-objekt. När den interna lagringsmängden är fylld skapas en större som får ta den fyllda mängdens plats och till vilken befintliga element överförs. (5p)

b) Skulle det vara lämpligt att låta `ObegränsadMängd` ärva `BegränsadMängd`? Båda är ju mängder men ändå olika. Argumentera antingen för eller emot, och redogör för vad de praktiska konsekvenserna då blir? (1p)



Figur 1: UML-diagram till uppgift 5.

5 Djur

UML-diagrammet i figur 1 på sidan 3 beskriver översiktligt några klasser för djur. Diagrammet är korrekt men viktiga detaljer kan ha utelämnats. Implementera det. Du får lägga till i diagrammet men inte ta bort/ändra redan befintligt. Dock får du inte lägga till fler klass- eller instansvariabler.

För alla djur gäller att de har ett namn och kan avge ett läte. Hundägare kan vara tvungna att betala hundskatt[1].

Djur Varje djur har ett namn som anges till konstruktorn och kan fås med metoden `namn()`. Metoden `prata()` returnerar en beskrivning av vad som hörs när djuret ger ett läte ifrån sig (pratar).

Katt Katter är djur som jamar när de pratar.

Hund (Fullvuxna) hundar är djur som skäller och vars ägare måste betala hundskatt. Skatten är just nu 120 kr/år men kan förstås ändras. Metoden `skatt()` returnerar skatten.

Valp Valpar är hundar som gnyr och för vilka skatten endast är 10% av den aktuella skatten för en (fullvuxen) hund.

NyföddValp En nyfödd valp är en valp för vilken skatten är 0 kr/år. (6p)

Kod till uppgift 1b

```
package a;
public class A {
  private int x;

  public A(int x) { this.x = x; }

  protected int get() {
    return x;
  }
}
```

```
package b;
import a.A;
public class B {
  A anObject = new A(3);

  void p() { System.out.println(anObject); }

  public static void main(String[] args) {
    new B().p();
  }
}
```

[1]Kuriosa: Åren 1923-1996 var det lag på hundskatt i Sverige.

substring (från klassen String i Javas standardbibliotek)

```
public String substring(int beginIndex, int endIndex)

Returns a new string that is a substring of this string. The
substring begins at the specified beginIndex and extends to
the character at index endIndex - 1. Thus the length of the
substring is endIndex-beginIndex.

Examples: "hamburger".substring(4, 8) returns "urge"
          "smiles".substring(1, 5) returns "mile"

Parameters:
    beginIndex - the beginning index, inclusive.
    endIndex - the ending index, exclusive.
Returns:
    the specified substring.
Throws:
    IndexOutOfBoundsException - if the beginIndex is negative,
    or endIndex is larger than the length of this String
    object, or beginIndex is larger than endIndex.
```

parseInt (från klassen Integer i Javas standardbibliotek)

```
public static int parseInt(String s) throws NumberFormatException

Parses the string argument as a signed decimal (base 10) integer.
The characters in the string must all be decimal digits, except
that the first character may be a minus sign ('-') or a plus
sign ('+').

Examples: Integer.parseInt("31") returns the integer 31.
          Integer.parseInt("-9") returns the integer -9.

Parameters:
    s - a String containing the int representation to be parsed
Returns:
    the integer value represented by the argument in decimal.
Throws:
    NumberFormatException - if the string does not contain a
    parsable integer.
```