DAT043 – Objektorienterad Programmering
Tentamen 2018-06-08
Tid: 08.30-12.30

Ansvarig lärare: Moa Johansson
Tfn: 031 772 10 78

Ansvarig lärare besöker tentamenssalarna ca klockan 9.30 samt 11.00.

Tentamensregler

Tentamen består av två delar. För att få **godkänt på tentan (betyg 3) måste man lösa minst fem av sju uppgifter i Del 1** (frågor numrerade 1–7). För att får högre betyg krävs att man utöver detta även löser uppgifter i Del 2 (frågor 8–9). För att få **betyg 4 ska, utöver godkänt på Del 1, även en fråga från Del 2 lösas**. För **betyg 5 skall, utöver godkänt på Del 1, dessutom båda frågorna i del 2 lösas**. Förutsatt att man klarat Del 1 får man självklart testa att lösa båda uppgifterna i Del 2. Lyckas man inte med någon har man ändå säkrat betyg 3, lyckas man bara med den ena så får man betyg 4. Minuspoäng ges ej.

Poäng på Del 2 kan *eventuellt* räknas mot godkänt om det skulle behövas (t.ex. fyra godkända lösningar på Del 1 och en godkänd lösning på Del 2 kan ge en trea i betyg), men det är osannolikt att man klarar frågorna i Del 2 om man inte samtidigt klarat Del 1.

Tillåtna hjälpmedel: Den tresidiga lathund som finns tillgänglig på kurssidan och upptryckt i ett exemplar till varje student vid tentamen (man ska alltså inte ta med sig en egen kopia).

Tentamensgranskning: Efter rättning finns tentamen tillgänglig på expeditionen på plan 4 i EDIT huset och kan granskas där. Önskar man diskutera rättningen kommer man kunna boka tid för detta genom att fylla i det formulär som kommer finnas länkat till från kurshemsidan efter att tentan är rättad. Notera att tentan i så fall ska lämnas kvar på expeditionen.

- Implementeringar ska skrivas i Java. Oväsentliga syntax- och namnfel eller liknande, betyder inte att lösningen underkänns. Det viktiga är att lösningar bedöms vara tillfredställande, d.v.s. att studenten med sitt svar tydligt visar förståelse för problemet i uppgiften och dess lösning.
- Skriv tydligt och välstrukturerat. Svårförståeliga lösningar kan underkännas.
- Lösningar som är onödigt krångliga eller inte följer god programmeringsstil, dels allmänt och dels med tanke på idéerna med objektorienterad programmering, kan underkännas.
- Om det inte uttryckligen står motsatsen i uppgiften kan du använda klasser och metoder i Java:s API.
- Om det inte uttryckligen står motsatsen i uppgiften får du definiera egna hjälpmetoder.
- Importeringar av klasser i Java:s API behöver inte skrivas ut.

Lycka till!

DEL 1

För att få godkänt på tentan (betyg 3) måste du **lösa minst fem av sju uppgifte**r i denna del.

Uppgift 1

Implementera en klass `Car` med tre instansvariabler: `model` (av typ `String`), `year` (av typ `String`) och `price` (av typ `double`). Klassen ska ha en konstruktor som initierar de tre instansvariablerna. Instansvariablerna ska *inte vara direkt tillgängliga utanför klassen* (använd en lämplig tillgänglighetsmodifierare som förhindrar detta). Implementera därför även `set` och `get` metoder för varje instansvariabel (t.ex. `setModel` och `getModel` osv.), som **uppdaterar** respektive **returnerar värdena** på de respektive instansvariablerna. Dessa metoder ska naturligtvis vara tillgängliga utanför klassen `Car`, så även här är det viktigt att ange en lämplig tillgänglighetsmodifierare.

Uppgift 2

(Du kan lösa den här uppgiften även om du inte har gjort Uppgift 1. Anta i så fall att klassen Car existerar enligt specifikationen i Uppgift 1.)

Skriv en klass `CarTest` som ska användas för att testa klassen `Car` från Uppgift 1. När man kör `CarTest` (med kommandot `java CarTest`) ska följande utföras:

- två olika objekt av typen `Car` ska skapas (du väljer själv initiala värden på objektens instansvariabler),
- därefter ska de två bilarnas priser skrivas ut (förslagsvis till terminalfönstret),
- slutligen ska bilarnas priser uppdateras: den första bilens pris ska rabatteras med 5% och den andra med 7%.

DEL 1

För att få godkänt på tentan (betyg 3) måste du **lösa minst fem av sju uppgifte**r i denna del.

Uppgift 3

Programmet nedan ska ta en URL-adress som argument och skriva ut HTML-koden till terminalfönstret.

```
import java.net.*;
import java.io.*;
import java.util.*;

public class Uppgift3{

  public static void main(String[] args) {
    String urlstr = args[0];
    URL url = new URL(urlstr);
    URLConnection conn = url.openConnection();
    BufferedReader in =
        new BufferedReader(new InputStreamReader(conn.getInputStream()));

     in
      .lines()
      .forEach(l -> System.out.println(l));
  }
}
```

Programmet kompilerar dock inte, och vi får följande felmeddelanden från kompilatorn:

```
javac Uppgift3.java

Uppgift3.java:9: error: unreported exception MalformedURLException; must be caught
or declared to be thrown
    URL url = new URL(urlstr);
              ^
Uppgift3.java:10: error: unreported exception IOException; must be caught or
declared to be thrown
    URLConnection conn = url.openConnection();
                                            ^
Uppgift3.java:11: error: unreported exception IOException; must be caught or
declared to be thrown
    BufferedReader in = new BufferedReader(new
InputStreamReader(conn.getInputStream()));

^
3 errors
```

Fixa programmet `Uppgift3` så att det kompilerar utan ovanstående fel.

Tips: `MalformedURLException` är en subklass till `IOException`.

DEL 1

För att få godkänt på tentan (betyg 3) måste du **lösa minst fem av sju uppgifte**r i denna del.

Uppgift 4

Studera koden nedan. Den innehåller ett interface `I`, samt tre klasser: `A`, `B` och `C`. Ange samtliga rader (numrerade 1–8) som **antingen orsakar ett runtime-fel när programmet körs, eller som orsakar kompileringsfel**. Som svar räcker det att ange den/de felaktiga raden/radernas siffra.

```
1)    A x1 = new A();
2)    I x2 = new C();
3)    A x3 = new B();
4)    B x4 = new C();
5)    B x5 = x1;
6)    C x6 = (C) x3;
7)    I x7 = new B();
8)    I x8 = (B) x3;
```

```
interface I{
  public int intMethod();
  public String strMethod();
}
```

```
class A{
   public String a;
}
```

```
class B extends A implements I{
  private int b;

  public int intMethod(){return b;}
  public String strMethod(){return a;}
}
```

```
class C extends A{
  private String c;

  public int intMethod(){return c.length()+a.length();}
  public String strMethod(){return a + c;}
}
```

DEL 1

För att få godkänt på tentan (betyg 3) måste du **lösa minst fem av sju uppgifte**r i denna del.

Uppgift 5

Vilka av följande påståenden **är sanna**?

A. Ett objekt i Java innehåller en egen ”kopia” av varje instansvariabel definierad i klassen som objektet tillhör.
B. Ett objekt i Java innehåller en egen ”kopia” av varje klassvariabel definierad i klassen som objektet tillhör.
C. Instansvariabler deklareras med nyckelordet `static`.
D. Variabler som deklareras med nyckelordet `final` kan inte uppdateras efter initiering.
E. Klassvariabler deklareras alltid med nyckelordet `final`.
F. Klassvariabler är globala variabler som ”delas” av alla objekt som hör till klassen.

Uppgift 6

Klassen `Pair` nedan är tänkt att representera par av objekt, som kan ha olika typ. Klassen använder därför typen `Object` för att representera de två objekten i paret. Ett bättre sätt att skriva en sådan klass är att använda sig av **Javas generics**. Skriv om klassen `Pair` nedan så att den använder sig av generics istället.

```
class Pair{
  Object fst, snd;
  public Pair(Object fst, Object snd) {
    this.fst = fst; this.snd = snd;
  }
}
```

DEL 1

För att få godkänt på tentan (betyg 3) måste du **lösa minst fem av sju uppgifte**r i denna del.

Uppgift 7

Studera klassen `Person` i rutan nedan.

```
public class Person{
  int age;
  String name;

  public Person(int a, String n){
    age = a;
    name = n;
  }
}
```

En testare skrivit ett testprogram för klassen `Person`:

```
public class Test{

  public static void main(String[] args) {
    String namn1 = "Karin Karlsson";
    String namn2 = "Pelle Persson";

    Person p1 = new Person(34, namn1);
    Person p2 = new Person(69, namn2);
    Person p3 = new Person(34, namn1);

    System.out.println("Test 1: Expects false got " + p1.equals(p2));
    System.out.println("Test 2: Expects true got " + p1.equals(p3));
  }
}
```

När man kör testprogrammet `Test` får man följande utskrift:

```
Test 1: Expects false got false
Test 2: Expects true got false
```

Uppenbarligen beter sig inte metoden `equals` som testaren förväntar sig i **Test 2**. Varför gör den inte det? Förklara kortfattat (i en eller ett par meningar) vad som händer. Fixa även klassen `Person` så testerna ovan returnerar de förväntade värdena.

DEL 2

Du behöver **bara svara på dessa frågor om du aspirerar på betyg 4 eller 5**. Vill du ha fyra ska du lösa en uppgift (välj själv) och för femma ska du lösa båda uppgifterna i den här delen, utöver de uppgifter du löst i Del 1.

Uppgift 8

Binärsökning är en sökalgoritm som vi stött på under kursens gång. Algoritmen förutsätter att arrayen är *sorterad*, och kan därför söka mer effektivt än en enkel linjär sökning. Man kan då genom att jämföra med elementet i mitten av arrayen avgöra i vilken halva som sökningen behöver fortsätta, och kan strunta i den andra halvan. Implementera följande metod **`binarySearch`** som letar efter ett element i en array med binärsökning.

- Metoden ska vara *generisk* och fungera för typer vilka implementerar interfacet `Comparable` (du ska alltså *inte* använda typen `Object` för argumenten utan ge argumenten generiska typer).
- Metoden ska ta två argument:
  - `x` är det element man söker efter.
  - `arr` är en array där elementen är sorterade i stigande ordning enligt `compareTo` metoden för elementtypen.
- Metoden ska returnera en `int` som är indexet i `arr` där `x` finns, eller `-1` om `x` inte finns i `arr`.

Vi påminner om att metoden `int compareTo(T arg)` returnerar:

- `0` om elementet som jämförs är lika med `arg`,
- ett heltal (`int`) mindre än 0 om elementet är mindre än `arg`,
- ett heltal större än 0 om elementet är större än `arg`.

OBS: Att använda sig av en färdig sökmetod från Javas bibliotek som `java.util.Arrays.binarySerach` ger naturligtvis **inte** rätt svar på denna fråga. *Du ska implementera algoritmen för binär sökning själv.*

Exempel på användning:

```
String[] myStrArr = {”a”, ”b”, ”c”, ”d”};
System.out.println(binarySearch(myStrArr, “c”);  // Skriver ut 2.

Integer[] myIntArr = {1, 2, 3, 4};
System.out.println(binarySearch(myIntArr, 5); // Skriver ut -1.
```

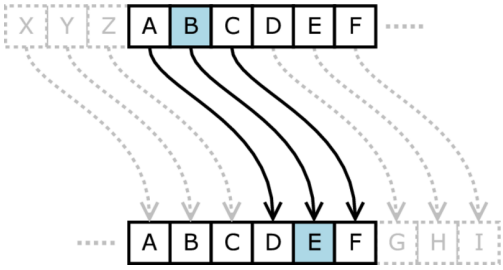
Tips: Studera interfacet `Comparable` i lathunden.

DEL 2

Du behöver **bara svara på dessa frågor om du aspirerar på betyg 4 eller 5**. Vill du ha fyra ska du lösa en uppgift (välj själv) och för femma ska du lösa båda uppgifterna i den här delen, utöver de uppgifter du löst i Del 1.

Uppgift 9

Ett Caesarshiffer är ett mycket enkelt sätt att kryptera meddelanden som går ut på att man har en heltalsnyckel **k**, och helt enkelt byter ut varje bokstav mot den bokstav som ligger **k** steg länge fram i alfabetet. Med exempelvis nyckeln **k=3**, blir texten ”abc” ---> ”def”. Om vi kommer till slutet av alfabetet börjar vi helt enkelt om från början, så med **k=3** blir ”ääö” ---> ”abc” (bilden nedan visar det engelska alfabetet som saknar ÅÄÖ, men principen är densamma).



Denna uppgift går ut på att skriva ett program `Uppgift9.java` som läser in en textfil, krypterar den med ett Caesarshiffer och skriver den krypterade texten till en annan fil.

För enkelhets skull antar vi att textfilen enbart innehåller ord innehållande bokstäver a-ö separerade med blanksteg (inga siffror, skiljetecken eller andra symboler). Du kan utgå från att alfabetet (strängen `ALPHABET` nedan) som programmet kan förväntas hantera redan är definierad i klassen `Uppgift9.java` (du behöver alltså inte skriva den själv):

```
public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyzääö";
```

Programmet ska inte skilja på stora och små bokstäver (d.v.s. ’A’ och ’a’ ska ses som samma bokstav) och den krypterade texten behöver bara innehålla små bokstäver. När man startar programmet ska man på kommandoraden ange tre argument: filnamnet för indatan, filnamnet för utdatan samt en (positiv) heltalsnyckel att använda i Caesarshiffret:

```
java Uppgift9.java Uppg9In.txt Uppg9Ut.txt 1
```

Om filen `Uppg9In.txt` innehåller texten:

```
Apa Katt Hund Örn
```

ska filen `Uppg9Ut.txt` efter kommandot ovan innehålla den krypterade texten:

```
bqb lbuu ivoe aso
```

Programmet ska kontrollera att antalet argument är korrekt, samt att filerna går att öppna. Om något skulle vara fel ska en felutskrift ges och programmet avslutas.

Tips: Skriv en separat hjälpmetod `String encrypt(String plainTxt, int key)` som krypterar en Sträng med en given nyckel först. Använd sen denna i huvudprogrammet.