Lösningar för tenta 2 DAT043, 2018-06-08.

Uppgift 1

```
public class Car{

  private String model;
  private String year;
  private double price;

  public Car(String model, String year, double price){
    this.model = model;
    this.year = year;
    this.price = price;
  }
  public String getModel(){return model;}
  public String getYear(){return year;}
  public double getPrice(){return price;}

  public void setModel(String m){model = m;}
  public void setYear(String y){year = y;}
  public void setPrice(double p){price = p;}
}
```

Uppgift 2

```
public class CarTest{

  public static void main(String[] args) {
    Car c1 = new Car("Volvo V40", "2014", 100000);
    Car c2 = new Car("Volvo Amazon", "1970", 5000);

    System.out.println("Pris 1: " + c1.getPrice());
    System.out.println("Pris 2: " + c2.getPrice());

    c1.setPrice(c1.getPrice() * 0.95);
    c2.setPrice(c2.getPrice() * 0.93);

  }
}
```

Uppgift 3

Kompilatorfelen beror på att vi inte har hanterat s.k. *checked exceptions*, som kan uppstå när man hanterar t.ex. I/O i Java. Kompilatorn kräver att man antingen deklarerar att metoden i fråga kan komma att kasta ett sådant undantag, eller att man lägger in koden i ett try-catch block som hanterar undantagen. Eftersom `MalformedURLException` är en subklass till `IOException` räcker det att man hanterar det senare för att få rätt svar på frågan (men att hantera dem båda separat godkänns givetvis också, och kan vara att föredra om man vill ge användaren mer information).

Alternativ 1: try-catch:

```
public static void main(String[] args) {
    String urlstr = args[0];

    try{
    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));
    }
    catch(IOException e){
      System.out.println("Uppgift3: " + e.getMessage());
    }
  }
```

Alternativ 2: med throws:

```
public static void main(String[] args) throws IOException{
…
```

Uppgift 4

2, 4, 5, 6 orsakar fel.

Frågan testar om man känner till hur arv (inheritance) fungerar i Java, samt hur interfaces används, och hur s.k. type-casts fungerar.

Uppgift 5

A, D, F är sanna.

Frågan testar om man känner till skillnaden mellan instansvariabler och klassvariabler. Instansvariabler finns det en instans (eller kopia om man så vill) för varje objekt som tillhör klassen. Instansvariabler tillhör alltså individuella objekt som skapats. Olika objekt som tillhör samma klass kan ha olika värden på sina instansvariabler. Klassvariabler däremot kan ses som global variabler, som ”delas” av samtliga objekt. Det finns alltså exakt en ”kopia” av dem och de deklareras med nyckelordet static. Nyckelordet final används för att deklarera att något är en konstant, dvs att de inte kan ändras efter initiering.

Uppgift 6

```
class Pair<T,U>{
  T fst;
  U snd;
  public Pair(T fst, U snd) {
    this.fst = fst; this.snd = snd;
  }
}
```

För denna fråga bör man använda två olika typ-variabler (som T och U ovan). Har man bara en typvariabel T går det bara att skapa par där både fst och snd är av samma typ. Dock har man fått rätt för ovanstående även om man använt samma typvariabel både för fst och snd då frågan inte explicit specificerade att fst och snd ska kunna ha olika typ.

Uppgift 7

Felet är att klassen Person inte har implementerat någon `equals`-metod. Eftersom alla klasser i Java implicit är subklasser till `Object`, så ärvs istället `equals`-metoden från `Object`. Denna kommer dock enbart att kontrollera om de två referenserna `p1` och `p3` är referenser till samma objekt, vilket de inte är! Om man vill kunna jämföra ifall objekt av typ `Person` har samma värden, så behöver man överskugga `equals`-metoden, så att objektens innehåll jämförs, istället för referensen.

Notera att man för att överskugga equals egentligen måste ha exakt samma typ som i equals-metoden för Object. Dock godkänns lösningar där argumentet är av typ Person.

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

  public Person(int a, String n){
    age = a;
    name = n;
  }
  public boolean equals(Object o){
    Person p = (Person) o;
    return age==p.age && name.equals(p.name);
  }
}
```

Uppgift 8

```
public static <T extends Comparable<T>> int binarySearch(T x, T[] arr) {
    int l = 0, r = arr.length - 1;
    while (l <= r) {
      int m = (l + r) / 2;
      int c = x.compareTo(arr[m]);
      if (c < 0) {  // x måste vara till vänster om m
          r = m - 1;
      } else if (c > 0) {  // x måste vara till höger om m
          l = m + 1;
      } else {  // c == 0, hittat x.
          return m;
      }
    }
    // hittade inte x
    return -1;
  }
```

Här gäller det att vara noga med att hålla reda på vilken del av arrayen man ska söka i. Det bästa är att definiera två variabler l och r som anger start och slut på delen man söker. Vi måste även ange att typen T ska vara sådan att det finns en compareTo-metod, vilket vi gör i metodens header.

Uppgift 9

Denna uppgift ska vara svår. Vi behöver använda oss av biblioteksfunktioner för klassen String, och då specifikt charAt och indexOf. Båda fanns med i lathunden som delades ut. utöver detta måste man naturligtvis även hålla rätt på vad man ska göra när man kommer till slutet av alfabetet.

Kod på nästa sida.

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

class Uppgift9{
  public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyzåäö";

  public static String encode(String text, int key){
    String plainTxt = text.toLowerCase();
    String cipher = "";
    for(int i = 0; i < plainTxt.length(); i++){
      int charPos = ALPHABET.indexOf(plainTxt.charAt(i));
      int keyVal = (key + charPos) % ALPHABET.length();
      char codeChar = ALPHABET.charAt(keyVal);
      cipher += codeChar;
    }
    return cipher;
  }

  public static void main(String[] args) {
    if (args.length != 3) {
          System.out.println("Felaktigt antal argument.");
          System.exit(0);
       }
    Scanner in = null;
    try{
      // Fil för indata.
      in = new Scanner(new File(args[0]));
     }
     catch(FileNotFoundException e){
      System.out.println(e.getMessage());
      System.exit(0);
    }
    PrintStream w = null;
    try{
      // Fil för utdata
    w = new PrintStream(new File(args[1]));
    }
    catch(FileNotFoundException e){
      System.out.println(e.getMessage());
      System.exit(0);
      }
      // Nyckel för kryptering.
    int key = Integer.parseInt(args[2]);

    while(in.hasNext()){
      String s = encode(in.next(),key);
      w.print(s + " ");
    }
    in.close();
    w.close();
  }

}
```