Anmälningskod:

Tentamen Programmeringsteknik I 2016-06-11

Skrivtid: 0900–1400

Tänk på följande

- Skriv läsligt. Använd *inte* rödpenna.
- Skriv bara på framsidan av varje papper.
- Lägg uppgifterna i ordning. Skriv uppgiftsnummer (gäller B-delen) och din kod *överst i högra hörnet* på alla papper
- Fyll i försättssidan fullständigt.
- Såvida inget annat anges, både får och ska man bygga på lösningar till föregående uppgifter även om dessa inte har lösts.
- På B-delen är det tillåtet att införa hjälpmetoder och hjälpklasser. Uttrycket ”skriv en metod som” skall alltså *inte* tolkas så att lösningen inte får struktureras med hjälp av flera metoder.
- Du behöver inte skriva `import`-satser för klasserna `Scanner`, `ArrayList`, `Locale` och inte heller för klasser i `java.io`.
- Alla uppgifter gäller programmeringsspråket Java och programkod skall skrivas i Java. Koden skall vara läslig dvs den skall vara vettigt strukturerad och indenterad. Namn på variabler, metoder, klasser etc skall vara beskrivande men kan ändå hållas ganska korta.

  Observera att betyget påverkas negativt av
  - icke-privata eller onödiga instansvariabler,
  - dålig läslighet,
  - upprepning av identisk kod,
  - underlåtenhet att utnyttja given eller egen tidigare skriven metod.

Skrivningen består av två delar. Lösningarna till uppgifterna på A-delen ska skrivas in i de tomma rutorna och den delen ska lämnas in. Lösningarna till uppgifterna på B-delen skrivs på lösa papper.

För att bli godkänd (betyg 3) krävs att hela A-delen är i stort sett rätt löst.

För betyget 4 krävs dessutom att minst hälften av uppgifterna på B-delen och betyg 5 att alla uppgifterna på B-delen är i stort sett rätt lösta. Vi bedömning av betyg 4 och 5 tas också hänsyn till kvalitéen på lösningarna i A-delen.

Observera att B-delen inte rättas om inte A-delen är godkänd.

Lycka till!

Anmälningskod:

Del A (obligatorisk för alla)

A1. Koden följer garanterat namnkonventionerna dvs klasser börjar med stor bokstav, metoder och variabler med liten.

Ringa in rätt svar och lämna in tillsammans med dina övriga svar!

a) En `void`-metod

1) får inte ha parametrar
2) får inte returnera ett värde
3) måste ha en utskriftsats

b) Givet koden

```
World w = new World();
Turtle t = new Turtle(w);
```

I denna kod är

1) `t` en klassvariabel
2) `t` ett objekt
3) `t` en referens till ett objekt

c) Koden

```
World w = new World();
Turtle t = new Turtle(w);
Turtle u = t;
u.move(10);
```

medför

1) att ett turtle-objekt flyttas
2) att två turtleobjekt flyttas
3) kompileringsfel
4) exekveringsfel

d) Vilken returtyp bör metoden nedan ha dvs vad ska ... bytas mot?

```
public ... calculate(int a) {
  return Math.sqrt(a*2) + 18;
}
```

1) `String`
2) `int`
3) `double`

e) Hur många objekt skapas av koden nedan? (Du ska bara räkna de objekt som man ser att koden skapar)

```
Turtle[] family = new Turtle[5];
World earth = new World(400, 400);
for(int i=0; i<3; i++) {
  family[i] = new Turtle(earth);
}
```

1) 3 objekt
2) 4 objekt
3) 5 objekt
4) 6 objekt
5) 7 objekt

Anmälningskod:

f) `Double` är

1) en inbyggd klass i Java
2) en metod för att hantera flyttal
3) en primitiv typ

g) Vilket uttalande om nedanstående kod är korrekt?

```
int[] tal = new int[5];
tal[0] = 14;
```

1) Den skapar en nollställd array och lägger in 14 först.
2) Den skapar en array där det första elementet blir 14 och resten null
3) Den skapar en array med *ett* element men kan växa till 5 element

h) Koden

```
Integer x = null;
System.out.println(x.toString());
```

1) ger kompileringsfel
2) ger `NullPointerException`
3) ger inga fel alls

i) `ArrayList` är en praktisk datastruktur för att

1) den kan växa när man vill lägga till fler element
2) man kan använda indexoperatorn `[ ]`
3) den kan lagra primitiva datatyper utan omslagsklasser

j) En instansvariabel

1) måste deklareras utanför konstruktorerna
2) måste deklareras i en konstruktor
3) måste ges ett värde i en konstruktor

k) Uttrycket `(int)(1 + 3/2.) + 3`

1) har värdet `5.0`
2) har värdet `5`
3) har värdet `5.5`

l) En konstruktor

1) skapar en ny klass
2) anropas alltid när ett objekt skapas
3) har returtypen `void`

Anmälningskod:

Nu följer ett antal uppgifter som handlar om klassen `Product`. Ett objekt ur denna klass representerar en enskild vara i en varuautomat. En vara representeras med ett namn, pris och antal exemplar som finns tillgängliga av den varan. Se bilagan!

A2. Skriv klart konstruktorn som tar emot namn och pris. Antalet `amount` sätts till 0.

```
public Product(String name, int  price) {
    this(name, price, amount);
}
```

A3. Skriv metoden `toString`. Ser körexemplet för det exakta utseendet!

```
public String toString()) {
    return name + ' ' + price + ' ' + amount;
}
```

A4. Skriv klart metoden `setPrice` som ger varan ett nytt pris. Metoden ska returnera det gamla priset.

```
public int setPrice(int newPrice) {
    int oldPrice = price;
    price = newPrice;
    return oldPrice;
}
```

A5. Skriv klart metoden `addAmount` som ökar på antalet i lager. Metoden ska returnera det nya antalet.

```
public int addAmount(int amount) {
    this.amount += amount;
    return this.amount;
}
```

Anmälningskod:

A6. Skriv klart `sell`. Om det finns exemplar av denna vara (`amount > 0`) ska antalet minskas med 1 och `true` returneras. I annat fall skall ingenting ändras och `false` returneras.

```
public boolean sell() {

    if (amount==0) {
        return false;
    } else {
        amount--;
        return true;
    }

}
```

Här följer ett antal uppgifter som handlar om klassen `VendingMachine`. Ett objekt ur denna klass representerar en varuatomat. Varorna representeras med en arraylist med `Product`-objekt. Förutom denna arraylist finns det två instansvariabler: `size` som anger det maximala antalet `Product`-objekt som kan lagras (dvs den största tillåtna storleken på arraylisten) och `cash` som anger aktuell kassa. Kassan är noll från början och ökar vid varje försäljning.

A7. Skriv klart konstruktorn som tar emot den maximalt tillåtna storleken, skapar arraylistan och sätter kassan till 0.

```
public VendingMachine(int size) {

    this.size = size;
    this.cash = 0;
    this.products = new ArrayList<Product>(size);

}
```

A8. Metoden `listProducts` ska producera en lista över varor som automaten normalt säljer. Varorna ska listas med namn och pris. Se körutskrift!

```
public void listProducts() {

    for (Product prod : products) {
      System.out.format("%d. %-8s %2d kr ",
                        prod.getName(), prod.getPrice());
    }

}
```

Anmälningskod:

A9. Metoden `find(String name)` letar efter en vara med angivet namn. Om varan hittas returneras en referens till den, i annat fall returneras `null`. Skriv klart metoden!

```
public Product find(String name) {
```

```
for (Product p: products) {
    if (p.getName().equals(name)) {
        return p;
    }
}
return null;
```

```
}
```

Nedanstående uppgift hör *inte* ihop med klasserna `Product` och `VendingMachine`.

A10. Metoden `random(int p)` ska med sannolikheten `p` procent returnerar `true` annars `false`.

Exempel: `random(100)` skall alltid returnera `true`, `random(0)` skall alltid returnera `false` och `random(90)` ska med 90 procents chans returnera `true` osv.

Skriv klart metoden!

```
public static boolean random(int p) {
```

```
return Math.random()*100 < p;
```

```
}
```

Del B (för betyg 4 och 5)

Svaren skrivs på lösa papper med ny uppgift på nytt papper.

B1. Metoden `addOrUpdate` i klassen `VendingMachine` letar efter en vara med angivet namn. Om varan hittas ändras priset och dess antal ökas på med angivet antal. Om varan inte hittas och det finns plats för en till läggs den till med angivet antal och pris. Om den ska läggas till men det inte finns plats görs ingenting. I så fall ska metoden returnera `false`, i övriga fall `true`.

```
public boolean addOrUpdate(String name, int price, int amount) {
  Product p = find(name);
  if (p!=null) {
    p.addAmount(amount);
    p.setPrice(price);
    return true;
  } else if (products.size()<size) {
    products.add(new Product(name, price, amount));
    return true;
  } else {
    return false;
  }
}
```

B2. Metoden `sell(String name)` ska hantera försäljning av en vara med angivet namn. Om varan inte finns alls eller om lagret av den är tomt (`amount` är 0) ska detta meddelas med en utskrift. Annars ska metoden skriva ut priset och anropa `cashReceived` med varans pris som parameter. Om `cashReceived` returnerar `true` ska ett leveransmeddelande skrivas ut. I annat fall ska ett meddelande om nekad leverans skrivas. Se utskrifterna!

Obs: Du ska inte skriva metoden `cashReceived` — bara anropa den.

```
public void sell(String name) {
  System.out.println("\nÖnskad vara: " + name);
  Product prod = this.find(name);
  if (prod==null || prod.getAmount()==0)
    System.out.println("*** " + name + " finns ej i lager");
  else {
    System.out.println("Betala " + prod.getPrice());
    if (cashReceived(prod.getPrice())) {
      cash += prod.getPrice();
      prod.sell();
      System.out.println(name + " levereras.");
    } else {
      System.out.println("*** Ingen betalning erhållen. Leverans avbruten.");
    }
  }
}
```

B3. Metoden `load` ska fylla på automaten med information från en fil. Filens format framgår av `save`-metoden. Om en vara inte kan läggas till (fler varukategorier än som det finns plats för) läggs den i stället in i en arraylist som returneras.

```
public ArrayList<Product> load(String filename) throws IOException
{
  Scanner scan = new Scanner(new BufferedReader(new FileReader(filename)));
  ArrayList<Product> result = new ArrayList<Product>();
  while (scan.hasNext()) {
    String name = scan.next();
    int price = scan.nextInt();
    int amount = scan.nextInt();
    if (addOrUpdate(name, price, amount)==false) {
      System.out.println("*** Can't load " + name);
      result.add(new Product(name, price, amount));
    }
  }
  scan.close();
  return result;
}
```

B4. Skriv en servicemetod `int readAnInt(String prompt, int min, int max)` som skriver ut promptern `prompt`, kontrollerar att det går att läsa ett heltal från standard input (`System.in`) samt kontrollerar att talet är större än eller lika med `min` och mindre är eller lika med `max`. Om detta är uppfyllt returneras talet. Om det inte är uppfyllt ska metoden ge en felutskrift och be användaren att försöka igen.

Så här kan konversationen se ut efter anropet `readAnInt("Gimme: ", 1, 5)`:

```
Gimme: 12
*** Not an integer between 1 and 5
*** Try again!
Gimme: -2
*** Not an integer between 1 and 5
*** Try again!
Gimme: Vad vill du ha egentligen?
*** Not an integer between 1 and 5
*** Try again!
Gimme: 9
*** Not an integer between 1 and 5
*** Try again!
Gimme: 1
```

```
public static int readAnInt(String prompt, int min, int max) {
    Scanner scan = new Scanner(System.in);
    while (true) {
         System.out.print(prompt);
         if (scan.hasNextInt()) {
              int res = scan.nextInt();
              if (res >= min && res <= max) {
                   return res;
              }
         }
         System.out.println("*** Not an integer between " + min + " and " + max);
         System.out.println("*** Try again!");
         scan.nextLine();
    }
}
```