TENTAMEN OOP 2017-08-05

ANVISNINGAR

Fråga 1 och 2 besvaras på det särskilt utdelade svarsbladet, övriga frågor på de vanliga svarsbladen. Du får gärna skriva på bägge sidorna, men påbörja varje uppgift på ett nytt blad. Vid inlämning ska svaren ligga sorterade efter frågornas nummer, inte efter den ordning du besvarat dem.

BETYGSÄTTNING

Tentan består av fem frågor, och varje fråga ger max 10 poäng. Maxpoängen på tentan är således 50. Betyget på tentan sätts enligt följande kriterier:

Fx – Minst 20 poäng på tentan. Betyget kan också ges om man får minst 25 poäng, men mindre än 8 poäng sammanlagt på fråga 1 och 2.

E – Minst 8 poäng sammanlagt på fråga 1 och 2, samt minst 25 poäng totalt.

D – Minst 8 poäng sammanlagt på fråga 1 och 2, samt minst 30 poäng totalt.

C – Minst 8 poäng sammanlagt på fråga 1 och 2, minst 35 poäng totalt, samt ingen uppgift med 0 poäng.

B – Minst 40 poäng totalt, samt ingen uppgift med 0 poäng.

A – Minst 45 poäng totalt.

HJÄLPMEDEL

Inga hjälpmedel förutom den syntaxsammanfattning som delas ut på tentan.

Lycka till!

Lösningsförslag läggs upp i ILearn senast tre arbetsdagar efter tentatillfället.

DEL A: FLERVALSFRÅGOR OCH KODFÖRSTÅELSE

Dessa frågor besvaras på det utdelade svarsbladet, inte på något annat sätt. På detta svarsblad får du bara skriva i de därtill avsedda fälten, ingen annanstans eftersom det delvis rättas maskinellt. Anteckningar och liknande utanför avsedda fält kan göra att det inte kan rättas, eller att resultatet blir fel. Sådana svar kommer inte att rättas om. Det är alltså ingen idé att skicka med anteckningar eller textuella förklaringar till dessa frågor eftersom vi bara kommer att titta på de avsedda fälten när vi rättar.

Flervalsfrågorna besvaras genom att fylla i hela rutan för alternativet med en mörk färg, så här:■, **inte** så här: ☒. Om du bara sätter ett kryss är risken att rättningsprogrammet missar att du markerat alternativet. Om du markerar ett felaktigt alternativ och vill ta bort det så fyller du i hela den kringliggande rutan. Skulle du göra fel två gånger på samma alternativ får du be tentavakten om ett nytt exemplar av svarsbladet och flytta över alla dina svar till det.

Exempel: A kommer att tolkas som rätt, B och D som fel. Hur C tolkas går inte att säga.

Om frågan frågar efter ”vilken” eller ”vilket” alternativ som är rätt så finns det exakt ett alternativ som är rätt. Skulle det av misstag från vår sida vara så att två alternativ är rätt så räcker det med att markera ett av dem, men det är mer troligt att du missat något som avgör vilket alternativ som egentligen är rätt.

Om frågan istället frågar efter ”vilka” alternativ som är rätt så kan noll, ett, eller flera av alternativen vara rätt, och samtliga ska vara markerade för att poäng ska ges. Man kan alltså inte få delpoäng på en sådan fråga.

FRÅGA 1 FLERVALSFRÅGOR

Denna fråga består av tio delfrågor av flervalstyp liknande de testfrågor som finns i Moodle. Varje delfråga ger noll eller en poäng.

FRÅGA 1A

Vilka av följande påståenden är sanna om parametrar till metoder?

A:	Har en datatyp
B:	Heter samma sak som klassen
C:	Om man skickar in ett objekt till en metod via en parameter så skapas en kopia på objektet
D:	Överskuggar variabler med samma namn på klassnivå

FRÅGA 1B

Ett program ska kontrollera mängden vatten i en vattentank som rymmer 1000 liter. Vilka av nedanstående boolska uttryck kan användas för att kontrollera om vattenmängden ligger inom det korrekta intervallet 0-1000?

A:	`v>=0 && v<=1000`	B:	`v<0 && v>1000`
C:	`v>=0 || v<=1000`	D:	`v<0 || v>1000`

FRÅGA 1C

Vilka av följande datatyper kan användas för att styra en switch-sats?

A: char	B: double	C: int	D: String

FRÅGA 1D

Om vi antar att Javas namngivningskonventioner följts, är då `createPlayList` namnet på en:

A: Klass	B: Konstant	C: Konstruktor	D: Metod

FRÅGA 1E

Vilken skyddsnivå bör instansvariabler normalt ha?

A:	**`public`**	B:	**`private`**
C:	**`public`** eller **`private`**, det spelar ingen roll	D:	Ingen, instansvariabler kan inte ha en skyddsnivå

FRÅGA 1F

Vilka av de fyra raderna kod nedan kompilerar?

A:	Double d = null;
B:	int i = null;
C:	if(str == null){ … }
D:	String metod(){ return null; }

(I alternativ C är str en String.)

FRÅGA 1G

Vilka värden får variablerna d och i efter att koden till höger körts?

```
double d=9.8765;
int i=(int)d;
d=(double)i;
```

A:	9.0 och 9	B:	9.0 och 10
C:	10.0 och 9	D:	10.0 och 10

FRÅGA 1H

Här nedanför finns fyra försök att simulera ett kast med en vanlig sexsidig tärning med hjälp av en slumpgenerator. Endast ett av dessa försök är korrekt, vilket?

A:	**`rnd.nextInt(6);`**	B:	**`rnd.nextInt(6+1);`**
C:	**`rnd.nextInt(6)+1;`**	D:	**`rnd.nextInt(7);`**

FRÅGA 1I

Vilket namn används för att referera till det egna objektet i en instansmetod?

A: Namnet på klassen	B: Object	C: self	D: this

FRÅGA 1J

Vad skrivs ut av följande kod?

```
for (int n = 0; n < 6; n++) {
   System.out.print(n % 3);
}
```

A:	012012
B:	012301
C:	120120
D:	123012

FRÅGA 2 KODFÖRSTÅELSE

Denna fråga består av två delfrågor där du ska komma fram till vad som skrivs ut när koden exekveras. Du skall i dina svar vara noga med vad som skrivs på vilken rad, alltså beakta skillnaden mellan print och println.

FRÅGA 2A

```
String[] a = { "7", "8", "9" };

for (int n = 1; n >= 0; n--) {
   for (int m = 0; m < 2; m++) {
      a[m] += a[n];
      System.out.print(a[m]);
   }
   System.out.println();
}
```

FRÅGA 2B

```
class A {

   private static int i;
   private int j;

   public A(int i) {
      this(i, i / 2);
   }

   public A(int i, int j) {
      A.i += i;
      this.j = j;
   }

   public String toString() {
      return "" + i + j;
   }

}
```

```
A a1 = new A(3);
System.out.println(a1);
A a2 = new A(2, 4);
System.out.println(a1);
System.out.println(a2);
```

DEL B: KODFRÅGOR

Denna del består av tre frågor där du själv ska skriva kod. Om inget annat sägs i själva frågan så spelar det ingen roll för betyget hur pass effektiv en lösning är så länge den fungerar och uppfyller de krav som ställs. Däremot ska grundläggande krav på god programmeringsstil följas – inom rimliga gränser. Förkortade namn och sent tillkomna inskjutna rader är alltså tillåtna.

FRÅGA 3

Denna uppgift går ut på att skriva en komplett klass som representerar ett kort med pengar på (tänk SL-kort med reskassa eller ett presentkort). Mängden pengar på kortet sätts när kortet skapas, och får inte vara lägre än noll. Om man gör ett försök att sätta det till ett lägre belopp så ska det istället sättas till noll.

Mängden pengar på kortet ska gå att läsa av. Det ska också gå att dra pengar från kortet. När man försöker dra pengar från kortet kan en av tre saker hända:

- Om beloppet man försöker dra är positivt och mindre eller lika med beloppet på kortet så dras beloppet från kortet och koden **`OK`** returneras.
- Om beloppet man försöker dra är positivt och större än beloppet på kortet så dras beloppet inte och koden **`ERR`** returneras.
- Om beloppet man försöker dra är noll eller lägre så dras beloppet inte och koden **`LOW`** returneras.

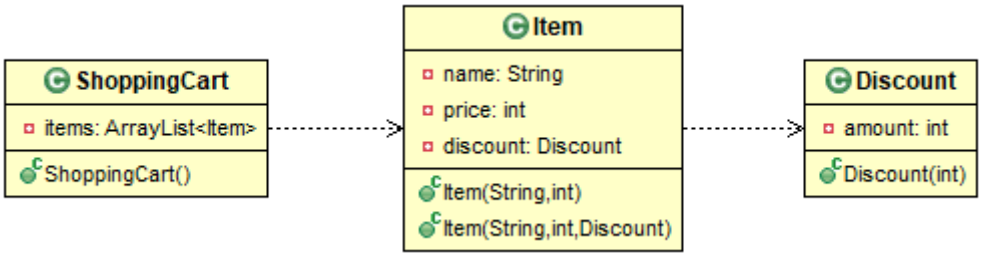
Felkoderna som returneras ska implementeras antingen som konstanter eller med hjälp av en uppräkningsbar typ (enum).

FRÅGA 4

Klassdiagrammet nedan visar några klasser som skulle kunna användas i ett system för en webb-butik: en kundvagnsklass, en varuklass och en rabattklass. Din uppgift är att skriva en metod till kundvagnsklassen som returnerar den totala kostnaden för varorna i kundvagnen med eventuella rabatter avdragna.

Samtliga attribut i klasserna är privata, så för att du ska kunna lösa uppgiften måste du lägga till metoder i de andra klasserna också.

För en vara som inte har någon rabatt är **`discount`** satt till **`null`**.



FRÅGA 5

Vid utskrift vill man ofta skriva ut en del fast text som aldrig förändras och en del som beror på värden i variabler. I exemplet nedan är det bara texten som är satt i fetstil som är föränderlig, resten är en mall som är samma varje gång.

> *Hej **Henrik**,*
>
> *Vi har ett otroligt erbjudande till dig på **strumpor** denna vecka. Svara omedelbart så får du **75** kronor rabatt.*
>
> *Med vänlig hälsning Affären AB*

Som ni säkert märkt när ni jobbat med inlämningsuppgifterna kan det bli ganska jobbigt att skriva ut denna typ av formulär med **`System.out.println`**-satser. Antingen blir det väldigt många utskriftsatser med små delar av texten i varje, eller så blir det många insprängda **`" + variabel + "`**. Oavsett vilket så blir det svårt att se sammanhanget i texten.

Det finns flera metoder i Java som kan hjälpa till med den här typen av utskrifter, till exempel **`String.format`** och **`System.out.printf`**, och din uppgift är nu att skriva en väldigt förenklad variant av den sistnämnda. Denna metod har nämnts på föreläsningar, och finns i boken, men ingår bara perifert på kursen, så allt som behövs för att lösa uppgiften förklaras nedan. Du behöver alltså inte känna till hur den riktiga **`printf`**-metoden fungerar.

Uppgiften går ut på att skriva en metod för utskrift som tar två parametrar: en sträng och en array av strängar. Den första parametern, strängen, innehåller texten som ska skrivas ut förutom de delar som ska variera. På dessa ställen i strängen står det istället ett procenttecken. Den andra parametern, arrayen, innehåller de strängar som dessa procenttecken ska bytas ut mot i samma ordning som de ska komma i strängen. Om metoden anropas med denna sträng:

```
"Hej %,\nVi har ett otroligt erbjudande till dig på % denna vecka. Svara omedelbart så får du % kronor rabatt.\nMed vänlig hälsning Affären AB"
```

Och denna array:

```
["Henrik", "strumpor", "75"]
```

Så skulle vi alltså få utskriften i exemplet ovan.

Tips: Uppgiften går utmärkt att lösa med de metoder som finns listade i syntaxsammanfattningen, men det finns ytterligare två metoder i klassen **`String`** som ni eventuellt kan ha nytta av:

```
String replaceFirst(String s, String replacement)
String[] split(String s)
```

Den sistnämnda metoden delar upp strängen på de platser **`s`** förekommer. Om vi till exempel har strängen **`"ABCDEBCFG"`** och anropar metoden med strängen **`"BC"`** kommer vi att få tillbaka en array med strängarna **`["A", "DE", "FG"]`**.

TENTAMEN OOP 2017-08-05

ANVISNINGAR

Fråga 1 och 2 besvaras på det särskilt utdelade svarsbladet, övriga frågor på de vanliga svarsbladen. Du får gärna skriva på bägge sidorna, men påbörja varje uppgift på ett nytt blad. Vid inlämning ska svaren ligga sorterade efter frågornas nummer, inte efter den ordning du besvarat dem.

BETYGSÄTTNING

Tentan består av fem frågor, och varje fråga ger max 10 poäng. Maxpoängen på tentan är således 50. Betyget på tentan sätts enligt följande kriterier:

Fx – Minst 20 poäng på tentan. Betyget kan också ges om man får minst 25 poäng, men mindre än 8 poäng sammanlagt på fråga 1 och 2.

E – Minst 8 poäng sammanlagt på fråga 1 och 2, samt minst 25 poäng totalt.

D – Minst 8 poäng sammanlagt på fråga 1 och 2, samt minst 30 poäng totalt.

C – Minst 8 poäng sammanlagt på fråga 1 och 2, minst 35 poäng totalt, samt ingen uppgift med 0 poäng.

B – Minst 40 poäng totalt, samt ingen uppgift med 0 poäng.

A – Minst 45 poäng totalt.

HJÄLPMEDEL

Inga hjälpmedel förutom den syntaxsammanfattning som delas ut på tentan.

Lycka till!

Lösningsförslag läggs upp i ILearn senast tre arbetsdagar efter tentatillfället.

DEL A: FLERVALSFRÅGOR OCH KODFÖRSTÅELSE

Dessa frågor besvaras på det utdelade svarsbladet, inte på något annat sätt. På detta svarsblad får du bara skriva i de därtill avsedda fälten, ingen annanstans eftersom det delvis rättas maskinellt. Anteckningar och liknande utanför avsedda fält kan göra att det inte kan rättas, eller att resultatet blir fel. Sådana svar kommer inte att rättas om. Det är alltså ingen idé att skicka med anteckningar eller textuella förklaringar till dessa frågor eftersom vi bara kommer att titta på de avsedda fälten när vi rättar.

Flervalsfrågorna besvaras genom att fylla i hela rutan för alternativet med en mörk färg, så här:■, **inte** så här: ☒. Om du bara sätter ett kryss är risken att rättningsprogrammet missar att du markerat alternativet. Om du markerar ett felaktigt alternativ och vill ta bort det så fyller du i hela den kringliggande rutan. Skulle du göra fel två gånger på samma alternativ får du be tentavakten om ett nytt exemplar av svarsbladet och flytta över alla dina svar till det.

Exempel: A kommer att tolkas som rätt, B och D som fel. Hur C tolkas går inte att säga.

Om frågan frågar efter ”vilken” eller ”vilket” alternativ som är rätt så finns det exakt ett alternativ som är rätt. Skulle det av misstag från vår sida vara så att två alternativ är rätt så räcker det med att markera ett av dem, men det är mer troligt att du missat något som avgör vilket alternativ som egentligen är rätt.

Om frågan istället frågar efter ”vilka” alternativ som är rätt så kan noll, ett, eller flera av alternativen vara rätt, och samtliga ska vara markerade för att poäng ska ges. Man kan alltså inte få delpoäng på en sådan fråga.

FRÅGA 1 FLERVALSFRÅGOR

Denna fråga består av tio delfrågor av flervalstyp liknande de testfrågor som finns i Moodle. Varje delfråga ger noll eller en poäng.

FRÅGA 1A

Vilka av följande påståenden är sanna om parametrar till metoder?

A:	Har en datatyp
B:	Heter samma sak som klassen
C:	Om man skickar in ett objekt till en metod via en parameter så skapas en kopia på objektet
D:	Överskuggar variabler med samma namn på klassnivå

FRÅGA 1B

Ett program ska kontrollera mängden vatten i en vattentank som rymmer 1000 liter. Vilka av nedanstående boolska uttryck kan användas för att kontrollera om vattenmängden ligger inom det korrekta intervallet 0-1000?

A:	`v>=0 && v<=1000`	B:	`v<0 && v>1000`
C:	`v>=0 || v<=1000`	D:	`v<0 || v>1000`

FRÅGA 1C

Vilka av följande datatyper kan användas för att styra en switch-sats?

A: char	B: double	C: int	D: String

FRÅGA 1D

Om vi antar att Javas namngivningskonventioner följts, är då `createPlayList` namnet på en:

A: Klass	B: Konstant	C: Konstruktor	D: Metod

FRÅGA 1E

Vilken skyddsnivå bör instansvariabler normalt ha?

A:	**`public`**	B:	**`private`**
C:	**`public`** eller **`private`**, det spelar ingen roll	D:	Ingen, instansvariabler kan inte ha en skyddsnivå

FRÅGA 1F

Vilka av de fyra raderna kod nedan kompilerar?

A:	Double d = null;
B:	int i = null;
C:	if(str == null){ … }
D:	String metod(){ return null; }

(I alternativ C är str en String.)

FRÅGA 1G

Vilka värden får variablerna d och i efter att koden till höger körts?

```
double d=9.8765;
int i=(int)d;
d=(double)i;
```

A:	9.0 och 9	B:	9.0 och 10
C:	10.0 och 9	D:	10.0 och 10

FRÅGA 1H

Här nedanför finns fyra försök att simulera ett kast med en vanlig sexsidig tärning med hjälp av en slumpgenerator. Endast ett av dessa försök är korrekt, vilket?

A:	`rnd.nextInt(6);`	B:	`rnd.nextInt(6+1);`
C:	`rnd.nextInt(6)+1;`	D:	`rnd.nextInt(7);`

FRÅGA 1I

Vilket namn används för att referera till det egna objektet i en instansmetod?

A: Namnet på klassen	B: Object	C: self	D: this

FRÅGA 1J

Vad skrivs ut av följande kod?

```
for (int n = 0; n < 6; n++) {
   System.out.print(n % 3);
}
```

A:	012012
B:	012301
C:	120120
D:	123012

FRÅGA 2 KODFÖRSTÅELSE

Denna fråga består av två delfrågor där du ska komma fram till vad som skrivs ut när koden exekveras. Du skall i dina svar vara noga med vad som skrivs på vilken rad, alltså beakta skillnaden mellan print och println.

FRÅGA 2A

```
String[] a = { "7", "8", "9" };

for (int n = 1; n >= 0; n--) {
   for (int m = 0; m < 2; m++) {
      a[m] += a[n];
      System.out.print(a[m]);
   }
   System.out.println();
}
```

SVAR

7888

7878887878

Viktigt att tänka på här är att arrayen består av strängar och inte tal.

FRÅGA 2B

```
class A {

   private static int i;
   private int j;

   public A(int i) {
      this(i, i / 2);
   }

   public A(int i, int j) {
      A.i += i;
      this.j = j;
   }

   public String toString() {
      return "" + i + j;
   }

}
```

```
A a1 = new A(3);
System.out.println(a1);
A a2 = new A(2, 4);
System.out.println(a1);
System.out.println(a2);
```

SVAR

31
51
54

DEL B: KODFRÅGOR

Denna del består av tre frågor där du själv ska skriva kod. Om inget annat sägs i själva frågan så spelar det ingen roll för betyget hur pass effektiv en lösning är så länge den fungerar och uppfyller de krav som ställs. Däremot ska grundläggande krav på god programmeringsstil följas – inom rimliga gränser. Förkortade namn och sent tillkomna inskjutna rader är alltså tillåtna.

FRÅGA 3

Denna uppgift går ut på att skriva en komplett klass som representerar ett kort med pengar på (tänk SL-kort med reskassa eller ett presentkort). Mängden pengar på kortet sätts när kortet skapas, och får inte vara lägre än noll. Om man gör ett försök att sätta det till ett lägre belopp så ska det istället sättas till noll.

Mängden pengar på kortet ska gå att läsa av. Det ska också gå att dra pengar från kortet. När man försöker dra pengar från kortet kan en av tre saker hända:

- Om beloppet man försöker dra är positivt och mindre eller lika med beloppet på kortet så dras beloppet från kortet och koden **`OK`** returneras.
- Om beloppet man försöker dra är positivt och större än beloppet på kortet så dras beloppet inte och koden **`ERR`** returneras.
- Om beloppet man försöker dra är noll eller lägre så dras beloppet inte och koden **`LOW`** returneras.

Felkoderna som returneras ska implementeras antingen som konstanter eller med hjälp av en uppräkningsbar typ (enum).

LÖSNINGSFÖRSLAG

```
//Denna uppgift går ut på att skriva en komplett klass som representerar ett kort
// med pengar på (tänk SL-kort med reskassa eller ett presentkort).
public class Card {

   private int balance;

   // Mängden pengar på kortet sätts när kortet skapas, och får inte vara lägre
   // än noll. Om man gör ett försök att sätta det till ett lägre belopp så ska
   // det istället sättas till noll.
   public Card(int balance) {
      if (balance < 0) {
         balance = 0;
      }
      this.balance = balance;
   }

   // Mängden pengar på kortet ska gå att läsa av.
   public int getBalance() {
      return balance;
   }
```

```
// Felkoderna som returneras ska implementeras antingen som konstanter eller
// med hjälp av en uppräkningsbar typ (enum).

// (och jo, ERR och LOW är riktigt dåliga namn. Jag ville ha något kort till
// tentan...)

public enum ErrorCode {
   OK, ERR, LOW
}

public static final int OK = 0;
public static final int ERR = 1;
public static final int LOW = 2;

// Det ska också gå att dra pengar från kortet.
// När man försöker dra pengar från kortet kan en av tre saker hända:
// • Om beloppet man försöker dra är positivt och mindre eller lika med
// beloppet på kortet så dras beloppet från kortet och koden OK returneras.
// • Om beloppet man försöker dra är positivt och större än beloppet på
// kortet så dras beloppet inte och koden ERR returneras.
// • Om beloppet man försöker dra är noll eller lägre så dras beloppet inte
// och koden LOW returneras.

// Denna version ändrar ordningen mellan alternativen för att få enklare
// villkor.
public ErrorCode withdrawVersion1(int amount) {
   if (amount <= 0)
      return ErrorCode.LOW;
   if (amount > balance)
      return ErrorCode.ERR;

   balance -= amount;
   return ErrorCode.OK;
}

// Samma version fast med konstanter istället för en uppräkningsbar typ
public int withdrawVersion2(int amount) {
   if (amount <= 0)
      return LOW;
   if (amount > balance)
      return ERR;

   balance -= amount;
   return OK;
}
```

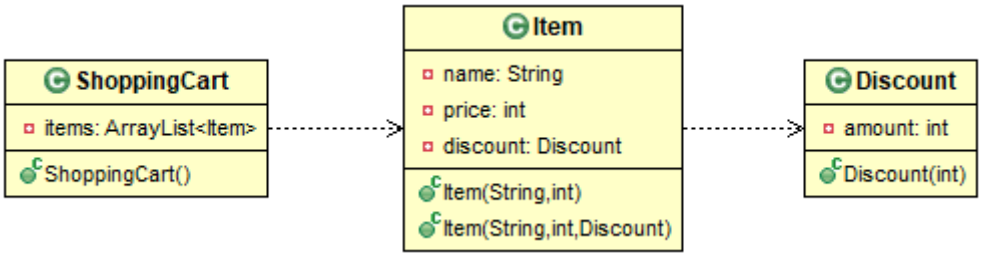
```
    // Om man tar villkoren rakt från frågan, så är det viktigt att det sista
    // villkoret inte skrivs ut explicit i en if-sats. Gör man det kommer
    // kompilatorn inte att inse att de tre fallen täcker alla möjligheter, och
    // koden kommer inte att kompilera.

    public ErrorCode withdrawVersion3(int amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            return ErrorCode.OK;
        } else if (amount > 0 && amount > balance) {
            return ErrorCode.ERR;
        } else {
            return ErrorCode.LOW;
        }
    }
}
```

FRÅGA 4

Klassdiagrammet nedan visar några klasser som skulle kunna användas i ett system för en webb-butik: en kundvagnsklass, en varuklass och en rabattklass. Din uppgift är att skriva en metod till kundvagnsklassen som returnerar den totala kostnaden för varorna i kundvagnen med eventuella rabatter avdragna.

Samtliga attribut i klasserna är privata, så för att du ska kunna lösa uppgiften måste du lägga till metoder i de andra klasserna också.

För en vara som inte har någon rabatt är **`discount`** satt till **`null`**.



LÖSNINGSFÖRSLAG VERSION 1: ENKLA GET-METODER

Klassen Discount behöver en get-metod för rabattbeloppet:

```
public int getAmount() {
    return amount;
}
```

Klassen Item behöver en möjlighet att läsa av såväl priset som rabatten. Detta kan göras med två get-metoder:

```
public int getPrice() {
   return price;
}

public Discount getDiscount() {
   return discount;
}
```

Nu kan vi använda dessa metoder för att räkna ut den totala kostnaden:

```
public int getTotalPriceVersion1(){
   int totalPrice=0;

   for(Item i: items){
      totalPrice+=i.getPrice();
      if(i.getDiscount()!=null){
         totalPrice-=i.getDiscount().getAmount();
      }
   }

   return totalPrice;
}
```

LÖSNINGSFÖRSLAG VERSION 2: EN HJÄLPMETOD

Ovanstående version fungerar, men skulle kunna göras lite tydligare om man inför ytterligare en metod i klassen Item:

```
public boolean hasDiscount() {
   return discount != null;
}
```

Metoden för den totala kostnaden blir då:

```
public int getTotalPriceVersion2(){
   int totalPrice=0;

   for(Item i: items){
      totalPrice+=i.getPrice();
      if(i.hasDiscount()){
         totalPrice-=i.getDiscount().getAmount();
      }
   }

   return totalPrice;
}
```

LÖSNINGSFÖRSLAG VERSION 3: EN BÄTTRE HJÄLPMETOD

Skillnaden mellan de två ovanstående versionerna är liten, och det kan diskuteras om metoden hasDiscount gör tillräcklig nytta för att vara motiverad. Om vi istället inför en metod i klassen Item för att räkna ut priset inklusive eventuell rabatt blir det mer intressant:

```
public int getPriceAfterDiscount() {
   if (hasDiscount()) {
      return price - discount.getAmount();
   } else {
      return price;
   }
}
```

Metoden för den totala kostnaden blir då betydligt enklare:

```
public int getTotalPriceVersion3() {
   int totalPrice = 0;

   for (Item i : items) {
      totalPrice += i.getPriceAfterDiscount();
   }

   return totalPrice;
}
```

ALTERNATIVA VERSIONER AV HJÄLPMETODERNA

De bägge hjälpmetoderna i Item innehåller varsin if-sats. Villkorsoperatorn ?: kan användas för att få kortare kod:

```
public int getDiscountAmountVersion2() {
   return hasDiscount() ? discount.getAmount() : 0;
}

public int getPriceAfterDiscountVersion2() {
   return hasDiscount() ? price - discount.getAmount() : price;
}
```

FRÅGA 5

Vid utskrift vill man ofta skriva ut en del fast text som aldrig förändras och en del som beror på värden i variabler. I exemplet nedan är det bara texten som är satt i fetstil som är föränderlig, resten är en mall som är samma varje gång.

*Hej **Henrik**,*

*Vi har ett otroligt erbjudande till dig på **strumpor** denna vecka. Svara omedelbart så får du **75** kronor rabatt.*

Med vänlig hälsning Affären AB

Som ni säkert märkt när ni jobbat med inlämningsuppgifterna kan det bli ganska jobbigt att skriva ut denna typ av formulär med **`System.out.println`**-satser. Antingen blir det väldigt många utskriftsatser med små delar av texten i varje, eller så blir det många insprängda **`" + variabel + "`**. Oavsett vilket så blir det svårt att se sammanhanget i texten.

Det finns flera metoder i Java som kan hjälpa till med den här typen av utskrifter, till exempel **`String.format`** och **`System.out.printf`**, och din uppgift är nu att skriva en väldigt förenklad variant av den sistnämnda. Denna metod har nämnts på föreläsningar, och finns i boken, men ingår bara perifert på kursen, så allt som behövs för att lösa uppgiften förklaras nedan. Du behöver alltså inte känna till hur den riktiga **`printf`**-metoden fungerar.

Uppgiften går ut på att skriva en metod för utskrift som tar två parametrar: en sträng och en array av strängar. Den första parametern, strängen, innehåller texten som ska skrivas ut förutom de delar som ska variera. På dessa ställen i strängen står det istället ett procenttecken. Den andra parametern, arrayen, innehåller de strängar som dessa procenttecken ska bytas ut mot i samma ordning som de ska komma i strängen. Om metoden anropas med denna sträng:

`"Hej %,\nVi har ett otroligt erbjudande till dig på % denna vecka. Svara omedelbart så får du % kronor rabatt.\nMed vänlig hälsning Affären AB"`

Och denna array:

`["Henrik", "strumpor", "75"]`

Så skulle vi alltså få utskriften i exemplet ovan.

Tips: Uppgiften går utmärkt att lösa med de metoder som finns listade i syntaxsammanfattningen, men det finns ytterligare två metoder i klassen **`String`** som ni eventuellt kan ha nytta av:

```
String replaceFirst(String s, String replacement)
String[] split(String s)
```

Den sistnämnda metoden delar upp strängen på de platser **`s`** förekommer. Om vi till exempel har strängen **`"ABCDEBCFG"`** och anropar metoden med strängen **`"BC"`** kommer vi att få tillbaka en array med strängarna **`["A", "DE", "FG"]`**.

LÖSNINGSFÖRSLAG VERSION 1: TECKEN FÖR TECKEN

Om vi begränsar oss till de av String:s metoder som tagits upp på kursen så är den enklaste versionen antagligen att skriva ut tecken för tecken och kontrollera om det är ett procenttecken:

```
public static void printfVersion1_0(String s, String[] args) {
   int iArgs = 0;
   for (int i = 0; i < s.length(); i++) {
      if (s.charAt(i) != '%') {
         System.out.print(s.charAt(i));
      } else {
         System.out.print(args[iArgs]);
         iArgs++;
      }
   }
}
```

Uppdateringen av iArgs kan flyttas in i indexeringen om man vill:

```
public static void printfVersion1_1(String s, String[] args) {
   int iArgs = 0;
   for (int i = 0; i < s.length(); i++) {
      if (s.charAt(i) != '%') {
         System.out.print(s.charAt(i));
      } else {
         System.out.print(args[iArgs++]);
      }
   }
}
```

OM NAMN PÅ KONSTANTER

Flera personer har deklarerat en konstant för procentttecknet på det här sättet:

```
private static final char PERCENT = '%';

public static void printfVersion1_2(String s, String[] args) {
   int iArgs = 0;
   for (int i = 0; i < s.length(); i++) {
      if (s.charAt(i) != PERCENT) {
         System.out.print(s.charAt(i));
      } else {
         System.out.print(args[iArgs++]);
      }
   }
}
```

Det här är en bra idé, men namnet är dåligt. Det säger inget mer än vad tecknet självt gör. Den här versionen är därför bättre:

```
private static final char REPLACEMENT_CHARACTER = '%';

public static void printfVersion1_3(String s, String[] args) {
   int iArgs = 0;
   for (int i = 0; i < s.length(); i++) {
      if (s.charAt(i) != REPLACEMENT_CHARACTER) {
         System.out.print(s.charAt(i));
      } else {
         System.out.print(args[iArgs++]);
      }
   }
}
```

LÖSNINGSFÖRSLAG VERSION 2: REPLACEFIRST

Med replacefirst från tipsen på tentan så kan vi göra en betydligt kortare version:

```
public static void printfVersion2(String s, String[] args) {
   for (String arg : args) {
      s = s.replaceFirst("%", arg);
   }
   System.out.print(s);
}
```

Ett vanligt fel hos dem som gjort denna variant är att man missat att uppdatera strängen. När det gäller strängar i Java så måste manalltid komma ihåg att metoderna inte ändrar på dem utan skapar en kopia.

LÖSNINGSFÖRSLAG VERSION 3: SPLIT

Bland tipsen nämndes också split-metoden. Denna metod låter väldigt bra när man först hör den, men den har ett stort problem i just det här fallet: Använder vi den så vet vi inte längre var procenttecknen stod. Om vi splittar ”A%B” och ”%A%B” på ”%”så kommer vi i bägge fallen få samma array: [”A”, ”B”]. Inget av nedanstående exempel fungerar alltså fullt ut, men de skulle bägge ge en hel del poäng på frågan ändå eftersom de stora dragen är där.

```
// Ger ett ArrayIndexOutOfBoudsException om s har detta utseende: x%y
public static void printfVersion3_1(String s, String[] args) {
   String[] parts = s.split("%");

   for (int i = 0; i < parts.length; i++) {
      System.out.print(parts[i] + args[i]);
   }
}

// Hoppar över sista delen om s har detta utseende: x%y
// Hoppar över sista delen om s har detta utseende: %x%y
// Ger ett ArrayIndexOutOfBoudsException om det finns flera % i följd
public static void printfVersion3(String s, String[] args) {
   String[] parts = s.split("%");

   for (int i = 0; i < args.length; i++) {
      System.out.print(parts[i] + args[i]);
   }
}
```