Anmälningskod:

Tentamen Programmeringsteknik I 2019-08-27

Skrivtid: 8:00 – 13:00

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 på 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`.
- All given kod följer kursens kodningsregler.
- 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. Rutorna är tilltagna i storlek så att de ska rymma svaren. En stor ruta betyder *inte* att svaret måste vara stort!

Lösningarna till uppgifterna på B-delen skrivs på lösa papper.

För att bli godkänd (betyg 3) krävs att minst ca 75% av 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 kvalitén 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. Ringa in rätt svarsalternativ eller skriv svar i ruta om sådan

a) Vad blir resultatet av följande kod?

```
double h = 1/10;
double s = 0;
for (int i= 1; i<=10; i++) {
   s += h;
}
```

1) Kompileringsfel
2) `RuntimeException`
2) `s` får värdet `0.0`
3) `s` får värdet exakt `1.0`
4) `s` får ett värde nära `1.0` men ej exakt

b) Satsen

```
double x = 4./2. + (int)(1. + 3/2);
```

resulterar i

1) Kompileringsfel
2) `x` får värdet `4.0`
3) `x` får värdet `4`
4) `x` får värdet `4.5`

c) Satsen

```
int x = 4./2. + (int)(1. + 3/2);
```

resulterar i

1) Kompileringsfel
2) `x` får värdet `4.0`
3) `x` får värdet `4`
4) `x` får värdet `4.5`

d) Hur många objekt skapas (synligt) av koden

```
Turtle[] a = new Turtle[6];
ArrayList<Turtle> t = new ArrayList<Turtle>();
t.add(new Turtle(new World());
```

e) Antag att nedanstående kod går att kompilera och köra

```
Test t = new Test();
System.out.println(t.ps[0].getName());
```

Markera ett alternativ som *måste* vara sant?

1) `ps` är en klassmetod
2) `ps` är en objektmetod
3) `ps` är en lokal variabel
4) `ps` är en formell parameter
5) `ps` är en array

f) Samma kod som i ovanstående uppgift. Vad bör gälla för variabeln `out`?

1) klassvariabel av primitiv typ
2) klassvariabel av referenstyp
3) instansvariabel av primitiv typ
4) instansvariabel av referenstyp
5) lokal variabel av primitiv typ
6) lokal variabel av referenstyp

g) Vad leder nedanstående kod till?

```
ArrayList<String> a = new ArrayList<String>();
a.add("Kalle");
a.add("Anka");
a.remove(0);
System.out.println(a.get(0));
```

1) Utskrift av `Kalle`
2) Utskrift av `Anka`
3) Utskrift av `null`
4) Utskrift av en tom sträng (`""`)
5) `NullPointerException`

För att denna uppgift ska betraktas som godkänd bör du ha minst 5 rätta deluppgifter.

Anmälningskod:

Resten av denna skrivning handlar om böcker i bokaffärer (”bokhandlar”).

Klassen `Book` representerar en bok med *titel*, *författare* och *antal* som anger hur många exemplar som finns inne.

Klassen `Store` representerar en bokhandel med *namn* och en *lista* med `Book`-objekt.

Klassen `AllStores` innehåller en lista över bokhandlar.

Dessutom finns en klass `Demo` med en `main`-metod som demonstrerar användningen av dessa klasser. Denna metod med dess utskrifter i högerspalten finns på nästa sida. Observera att utskrifterna från några av `toString`-metoderna är avklippta.

Läs igenom koden och utskrifterna så att du förstår hur det ska fungera! Dina metoder ska ge exakt samma resultat som körexemplen visar!

När du löser uppgifterna kan du förutsätta att klassen `Store` innehåller en `toString` som fungerar enligt utskrifterna från demoprogrammet samt en `getName`-metod som returnerar namnet på butiken.

```
public static void main(String[] args) {
   System.out.println("Demo of Book\n" +          Demo of Book
                      "============");           ============
   Book b = new Book("La Peste", "Camus", 1);
   b.addCopies(2);
   System.out.println(b);                         <Camus: La Peste(3)>
   System.out.print("\"" + b.getTitle() +
                    "\" by " + b.getAuthor());
   System.out.println(" is available in " +
                      b.getNumber() +
                      " copies");                 "La Peste" by Camus is available in 3 copies
   b.removeCopy();
   System.out.println(b);                         <Camus: La Peste(2)>
   b.removeCopy();
   System.out.println(b);                         <Camus: La Peste(1)>
   b.removeCopy();
   b.removeCopy();                                *** No copies available: <Camus: La Peste(0)>
   System.out.println(b);                         <Camus: La Peste(0)>

   System.out.println("\nDemo of Store\n" +       Demo of Store
                      "=============");           =============
   Store lundeq = new Store("Lundeq");
   lundeq.addBook("Kallocain", "Boye", 2);
   lundeq.addBook("Iliaden", "Homeros", 1);
   lundeq.addBook("Aniara", "Martinson", 1);
   lundeq.addBook("Iliaden", "Homeros", 2);
   lundeq.addBook("Ulysses", "Joyce", 1);
   lundeq.addBook("A Farewell to Arms",
                  "Hemingway", 3);
   System.out.println(lundeq.toString());         Lundeq: [<Boye: Kallocain(2)>, <Homeros: Iliaden(3)>, <Martinson:
   System.out.println(
            "Total number of books: " +
            lundeq.totalNumberOfBooks());         Total number of books: 10
   System.out.println(
            "Most common book    : " +
             lundeq.mostCommonBook());            Most common book    : <Homeros: Iliaden(3)>

   lundeq.sellBook("Kallocain");                  Sold: <Boye: Kallocain(1)>
   lundeq.sellBook("Iliaden");                    Sold: <Homeros: Iliaden(2)>
   lundeq.sellBook("Pesten");                     Sorry! Unknown book: Pesten
   lundeq.sellBook("Kallocain");                  Sold: <Boye: Kallocain(0)>
   System.out.println(lundeq.toString());         Lundeq: [<Boye: Kallocain(0)>, <Homeros: Iliaden(2)>, <Martinson:
   lundeq.sellBook("Kallocain");                  Sorry! Out of stock: Kallocain
   lundeq.print();                                Lundeq:
                                                     <Boye: Kallocain(0)>
                                                     <Hemingway: A Farewell to Arms(3)>
                                                     <Homeros: Iliaden(2)>
                                                     <Joyce: Ulysses(1)>
                                                     <Martinson: Aniara(1)>

   System.out.println("\nDemo of AllStores\n" +  Demo of AllStores
                      "=================");       =================
   Store almq = new Store("Almq");
   almq.addBook("Iliaden", "Homeros", 3);
   almq.addBook("Kris", "Boye", 1);
   AllStores stores = new AllStores();
   stores.addStore(lundeq);
   stores.addStore(almq);

   stores.searchBook("Iliaden");                  Searching for Iliaden
                                                      Lundeq has 2 copies.
                                                      Almq has 3 copies.
   stores.searchBook("Kris");                     Searching for Kris
                                                      Almq has 1 copies.
   stores.searchBook("Hamlet");                   Searching for Hamlet
}                                                     Sorry! Nothing found.
```

Anmälningskod:

A2. Deklarera instansvariablerna `title`, `author` och `number` (med uppenbara betydelser) i klassen `Book`:

A3. Skriv färdigt nedanstående konstruktor

```
public Book(String title, String author, int number) {
```

```
}
```

A4. Skriv metoderna `getAuthor`, `getTitle` och `getNumber` i klassen `Book`.

Anmälningskod:

A5. Skriv `toString`-metoden i klassen `Book`! Se demoprogrammet!

A6. Skriv klart metoden `addCopies` i klassen `Book`. Metoden ska öka antalet exemplar med givet parametervärde.

```
public void addCopies(int number) {
```

```
}
```

A7. Skriv metoden `removeCopy()` i klassen `Book`. Metoden ska minska antalet kopior med 1. Om antalet kopior redan är 0 ska en felutskrift ges och antalet lämnas oförändrat (0).

Anmälningskod:

A8. Klassen `Store` ska ha en instansvariabel `name` som lagrar namnet på bokhandeln och en `theBooks` som är en arraylista med `Book`-objekt.

Deklarera dessa!

A9. Skriv en konstruktor `public Store(String name)`

A10. Metoden `searchBook` i klassen `Store` letar efter en bok med angiven titel (oavsett författare) och returnerar en referens till den. Om boken inte hittas returneras `null`.

Skriv klart metoden

```
public Book searchBook(String title) {
```

```
}
```

Anmälningskod:

A11. Skriv metoden `public int totalNumberOfBooks()` i klassen `Store`. Metoden ska räkna och returnera det totala antalet exemplar av alla böcker som finns i butiken.

A12. Skriv metoden `public Book mostCommonBook()` i klassen `Store` som returnerar det bokobjekt som har flest antal exemplar.

Del B (för betyg 4 och 5)

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

B1. Skriv metoden `void addBook(String title, String author, int number)` i klassen `Store`. Om en bok med den titeln redan finns lagrad ska dess antalsangivelse uppdateras med *number*. Om den inte finns lagrad ska den läggas till sist i listan över böcker.

Tips: Använd metoden `searchBook`!

B2. Skriv metoden `public void sellBook(String title)` i klassen `Store`. Se demoprogrammet för funktion!

B3. Skriv metoden `public void print()` i klassen `Store` som listar alla `Book`-objekt bokstavsordning efter författare.

B4. Skriv klassen `AllStores` som samlar ett antal `Store`-objekt och möjliggör sökning över alla butiker. Se, som vanligt, demoprogrammet för nödvändiga metoder och funktioner.