

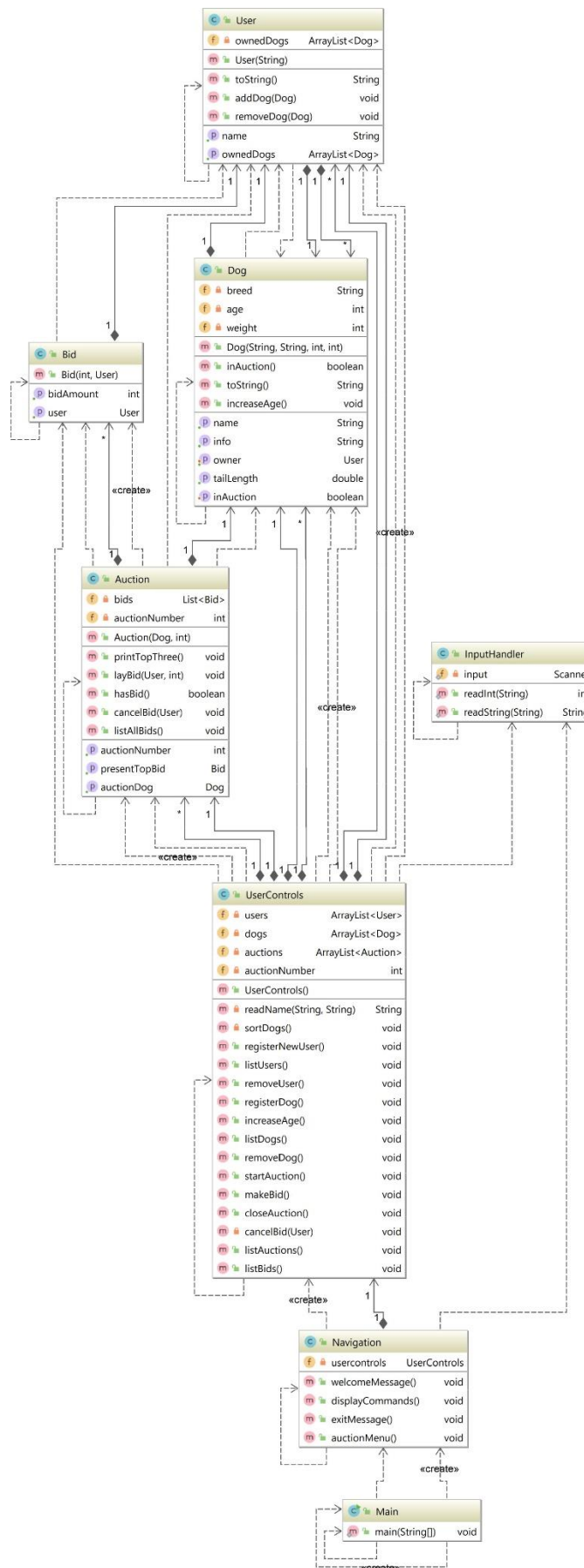
# PROG1 HT18 inlämningsuppgift

Grupp A eller B: A  
Gruppnummer: 401

Förnamn	Efternamn	Användarnamn
Markus	Bowie	mabo1371

Eventuella kommentarer om samarbete kan skrivas i fältet nedan

# 1 KLASSDIAGRAM SLUTGILTIG VERSION<sup>1 2</sup>



Samma som det redan godkända.

## 2 STILKRAVEN<sup>3</sup>

---

- ✓ Java 8
- ✓ Egna filer

Jag har delat upp programmet i 8 filer. Detta för att enklare kunna resonera kring programmet i delar och, även om det inte är aktuellt i detta fall, för enklare uppskalning av systemet.

- ✓ Inga paket
- ✓ UTF-8 utan BOM
- ✓ Information om vilka som samarbetat
- ✓ Indentering
- ✓ Namngiving

Jag har valt variabel- och klassnamn som ska ge utvecklaren en snabb idé om dess innehåll. För lokala variabler i loopar har jag använt variabelnamn såsom "i", då dessa enbart är tillfälliga och används lokalt i loopen.

- ✓ Privata data och genomtänkta skyddsnivåer på metoder

Jag har försökt begränsa åtkomsten mellan metoder till sådant som är nödvändigt för programmets funktion. Att ha färre publika metoder underlättar för mig som utvecklare att bilda mig en uppfattning om hur klassen ska användas.

- ✓ Enbart motiverade statiska metoder och variabler

Jag skriver om denna punkt i del 6.

---

```
C UserControls
m UserControls()
m registerNewUser(): void
m listUsers(): void
m removeUser(): void
m registerDog(): void
m increaseAge(): void
m listDogs(): void
m removeDog(): void
m startAuction(): void
m makeBid(): void
m closeAuction(): void
m listAuctions(): void
m listBids(): void
m readName(String, String): String
m sortDogs(): void
m cancelBid(User): void
```

### 3 KORTA METODER OCH LITE KODUPPREPNING<sup>4</sup>

---

RegisterNewUser-metoden är ett exempel på en kort metod. I synnerhet eftersom den anropar readName-metoden.

```
65 public void registerNewUser() {
66     String name = readName( prompt: "Enter the name of the user: ", type: "name");
67
68     User u = new User(name);
69     users.add(u);
70
71     System.out.println(name + " added to the register");
72
73 }
```

Ett bra exempel på något jag implementerat för att minimera kodupprepning är InputHandler-klassen. Alla metoder i programmet som kräver en Scanner gör sina anrop hit.

```
1  //.../
4
5  import java.util.Scanner;
6
7  public class InputHandler {
8
9      private static Scanner input = new Scanner(System.in);
10
11     public static int readInt(String prompt) {
12         System.out.println(prompt);
13         int no = input.nextInt();
14         input.nextLine();
15         return no;
16     }
17
18     @ public static String readString(String prompt) {
19         System.out.println(prompt);
20         return input.nextLine().trim().toLowerCase();
21     }
```

## 4 ARRAYANVÄNDNING<sup>5</sup>

---

Jag använder Arrayer i User-klassen.

```
1  + //.../
4
5  import java.util.ArrayList;
6
7  public class User {
8
9      private String name;
10
11     private Dog[] ownedDogs = new Dog[0];
12
13     + public User(String name) { this.name = name; }
16
17     + public String getName() { return name; }
20
21     - public ArrayList<Dog> getOwnedDogs() {
22         ArrayList <Dog> dogArrayList = new ArrayList<>();
23         - for (Dog ownedDog : ownedDogs) {
24             dogArrayList.add(ownedDog);
25         }
26         return dogArrayList;
27     }
28 }
```

Hade jag fått välja hade jag använt mig av ArrayList i hela programmet för att den undviker en massa manuellt arbete med arrayer. Då uppgiften krävde att jag skulle använda mig av arrayer på en plats gjorde jag det. (Det var dock lärorikt!)

## 5 EGENIMPLEMENTERAD SORTERING<sup>6</sup>

```
28 private void sortDogs() {
29     for (int i = 0; i < dogs.size() - 1; i++) {
30         Dog leftDog = dogs.get(i);
31         Dog rightDog = dogs.get(i + 1);
32         String leftName = leftDog.getName();
33         String rightName = rightDog.getName();
34         double leftTail = leftDog.getTailLength();
35         double rightTail = rightDog.getTailLength();
36         if (Double.compare(leftTail, rightTail) > 0) {
37
38             dogs.set(i, rightDog);
39             dogs.set(i + 1, leftDog);
40
41             i = -1;
42
43         }
44
45         if (Double.compare(leftTail, rightTail) == 0) {
46
47             if (leftName.compareTo(rightName) > 0) {
48
49                 dogs.set(i, rightDog);
50                 dogs.set(i + 1, leftDog);
51
52                 i = -1;
53
54             }
55
56         }
57     }
58 }
59 }
```

Jag valde att använda mig av bubble sort eftersom jag pratade om den med några i skolan och den verkade intressant och relativt lättförståelig. Jag läste framförallt på om den på wikipedia:

[https://en.wikipedia.org/wiki/Bubble\\_sort](https://en.wikipedia.org/wiki/Bubble_sort)

## 6 ALL FEEDBACK FRÅN TESTPROGRAMMET + ERA KOMMENTARER<sup>7</sup>

### Generellt omdöme

Det mesta ser okej ut, men det finns saker kvar som du bör titta på.

### Kodanvändning och stil

Nedanstående metod(er) är på gränsen till att börja bli lång(a). Det här behöver inte vara ett problem, men du bör titta på dem och fundera på om de verkligen utför en enda uppgift, och inte bör delas upp i mindre.

- \* [Navigation.java:24](#): auctionMenu (44 programsatser)
- \* [UserControls.java:153](#): removeDog (16 programsatser)
- \* [UserControls.java:181](#): startAuction (22 programsatser)
- \* [UserControls.java:213](#): makeBid (25 programsatser)
- \* [UserControls.java:255](#): closeAuction (18 programsatser)

Nyckelordet static används fler gånger i koden än vad som förväntas. Det här behöver inte vara ett fel, men du bör titta en extra gång på de ställen där du gör det och ta dig en funderare på om användningen verkligen är motiverad på alla ställen. Normalt är det bara main-metoden och kanske någon enstaka konstant som är statisk i denna uppgift.

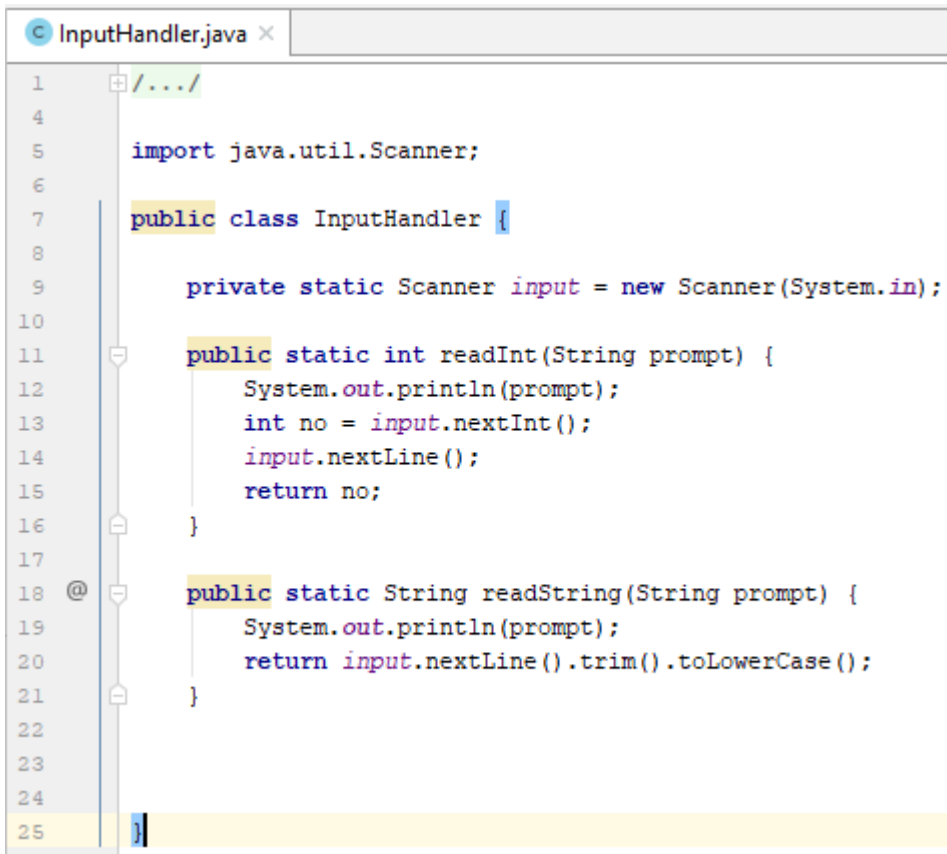
- \* 4 static found

### Mått

Här är några mått som eventuellt kan vara av intresse.

# classes: 8  
# methods: 46  
# statements: 347

Under handledning fick jag höra att jag använde mig av för många Scanners. Då valde jag att sätta alla i en egen klass istället. Alternativet att skicka runt samma input handler överallt verkade lite överdrivet för det här projektet.



```
1  // ...
4
5  import java.util.Scanner;
6
7  public class InputHandler {
8
9      private static Scanner input = new Scanner(System.in);
10
11     public static int readInt(String prompt) {
12         System.out.println(prompt);
13         int no = input.nextInt();
14         input.nextLine();
15         return no;
16     }
17
18     @
19     public static String readString(String prompt) {
20         System.out.println(prompt);
21         return input.nextLine().trim().toLowerCase();
22     }
23
24
25 }
```

Vad gäller Main så måste det vara statiskt:



```
1  .../
4
5  import java.util.ArrayList;
6
7
8  public class Main {
9
10
11  public static void main(String[] args) {
12
13
14      Navigation mainObject = new Navigation();
15      Main mainProgram = new Main();
16      mainObject.welcomeMessage();
17      mainObject.auctionMenu();
18      mainObject.exitMessage();
19
20  }
21
22
23 }
24
```



---

<sup>1</sup> Alla avsnitt ska vara numrerade och i den ordning som mallen listar dem så att vi lätt kan hitta dem. Ni ska inte ha någon innehållsförteckning, eller lägga till några andra saker som inte efterfrågas. Alla avsnitt börjar på en ny sida.

<sup>2</sup> Klassdiagrammet kan se väldigt litet ut här, men ska vara läsbart i pdf-filen om man förstorar den.

Om det slutgiltiga klassdiagrammet är identiskt med det som lämnades in och blev godkänt behöver ni inte göra något mer här, annars måste ni lista, och vid behov motivera, de ändringar som gjorts så att vi kan bedöma om det fortfarande är godkänt. Motivationerna ska vara kortfattade, men det räcker inte med att säga "vi behövde" eller "för att få det att fungera", utan det ska vara en riktig motivation.

<sup>3</sup> Detta avsnitt ska visa hur ni tänkt kring stilkraven på uppgiften, framförallt kraven på namngivning, genomtänkta skyddsnivåer på metoder och enbart motiverade statiska metoder och variabler. Avsnittet ska vara kort, max en sida, och helst betydligt kortare än så. Behövs det mer än några få meningar för att förklara hur ni tänkt på någon av ovanstående punkter är det troligt att ni har ett problem i er kod. Ta gärna med korta exempel från koden, de brukar hjälpa förståelsen.

Precis om på föregående punkt ska motivationerna vara ordentliga. Att till exempel säga "Vi valde namn som vi tyckte var bra", eller " är inte intressant.

<sup>4</sup> Detta avsnitt ska visa hur ni tänkt för att uppfylla kraven på korta metoder och lite kodupprepning. Formatet är fritt, men det är troligtvis bäst att utgå från några exempel i koden och kort förklara hur ni har delat upp en lång metod i mindre logiska delar eller återanvänt kod på flera ställen.

<sup>5</sup> Detta avsnitt ska tydligt visa att ni uppfyllt kravet på användning av arrayer. Ett lämpligt format är ett par meningar om var ni valt att använda arrayer och varför, samt utdrag ur koden som tydligt visar att ni faktiskt använder arrayen ordentligt.

Att vi inte frågar om användning av ArrayList är inte ett fel i mallen. Det beror på att alla alltid använder den klassen eller någon annan ur samlingsbiblioteket.

<sup>6</sup> Detta avsnitt ska tydligt visa att ni uppfyllt kravet på en egenimplementerad sortering. Ett lämpligt format är ett par meningar om vilken sorteringsalgoritm ni valt och var ni läst på om den, samt den fullständiga koden för sorteringen.

<sup>7</sup> Detta avsnitt ska lista all feedback som det automatiska testprogrammet ger med era kommentarer och, vid behov, motivationer till denna feedback. Det enklaste sättet att få tag på feedbacken är via Edit-fliken. Motivationerna ska vara kortfattade, men det räcker inte med att säga "vi tyckte det var okej" eller "för att få det att fungera", utan det ska vara riktiga motivationer.

Följande behöver inte kommenteras eller motiveras:

- Varningar för att metoder börjar bli för långa, om det inte är väldigt många sådana varningar.
- Varningen för att metoden som kontrollerar kommandonamnen är för lång. Detta är troligtvis okej, om den inte är jättelång.
- Uppenbart ointressanta kommentarer. Den här punkten är naturligtvis subjektiv, och det är den som rättar som har tolkningsföreträde, så är ni tveksamma till om något behöver kommenteras eller motiveras så gör det, eller fråga en handledare.