1DV506, Problemlösning och programmering, ht 2017

Laboration 3

Problem? Tveka inte att fråga er labbhandledare (eller Lars på lektionerna) om ni har problem med någon av uppgifterna. Ni kan också använda forumet i Moodle.

Eclipse förberedelser: Skapa ett nytt paket med namnet `DittLnuAnvändarNamn_lab3` inne i Java-projektet `1DV506` och spara alla filer relaterade till denna lab i det paketet.

Lektion 6 - Metoder

I uppgifterna 1 och 2 skall ni skapa ett antal statiska metoder. De skall alltid skapas inuti samma klass som innehåller main-metoden. Main-metoden skall fungera som ett testprogram som demonstrerar hur de olika metoderna kan användas.

- **Uppgift 1**
  Skriv en klass `Arrays.java` som förutom main-metoden innehåller följande **statiska** metoder:
  - Metoden `int sum(int[] arr)` som adderar ihop alla element i arrayen `arr` och returnerar summan.
  - Metoden `String toString(int[] arr)` som bygger upp en sträng som när den skrivs ut ger en snygg utskrift av arrayens innehåll. Den skall kunna användas på följande sätt med gott resultat

    ```
    int[] n = {3,4,5,6,7};
    String str = Arrays.toString(n);
    System.out.println("n = " + str);
    ```

  - Metoden `int[] addN(int[] arr, int n)` som bygger upp, och returnerar, en ny array där man adderat talet `n` till alla element i arrayen `arr`. Arrayen `arr` skall lämnas opåverkad.
  - Metoden `int[] reverse(int[] arr)` som bygger upp, och returnerar, en ny array där alla element i arrayen `arr` finns i omvänd ordning. Arrayen `arr` skall lämnas opåverkad.
  - Metoden `boolean hasN(int[] arr, int n)` som returnerar `true` om `n` finns i arrayen `arr`, annars `false`.
  - Metoden `void replaceAll(int[] arr, int old, int nw)` som byter ut alla förekomster av `old` mot `nw` i `arr`.
  - Metoden `int[] sort(int[] arr)` som returnerar en ny sorterad array (minsta först) med samma mängd av heltal som `arr`. Arrayen `arr` skall lämnas opåverkad.
  - Metoden `boolean hasSubsequence(int[] arr, int[] sub)` som returnerar `true` om subsekvensen `sub` finns i arrayen `arr`, annars `false`. I fallet `hasSubsequence({1,2,3,4,5}, {3,4,5})` skall den alltså returnera `true` eftersom `{3,4,5}` finns som sista del i `{1,2,3,4,5}`.

  Notera: Ni förväntas att implementera alla dessa metoder utan att använda någon av de array-relaterade hjälpklasserna i Javas klassbibliotek.

- **Uppgift 2**
  Skapa en klass `Pnr.java` som förutom `main`-metoden innehåller ett antal statiska metoder relaterade till svenska personnummer. Antag att alla personnummer är en sträng på formen `ÅÅMMDD-NNNN`. Det skall finnas följande medlemmar i klassen.

1. Metoderna `getFirstPart` och `getSecondPart` som returnerar personnumrets första (ÅÅMMDD) och andra (NNNN) del.
2. `isFemaleNumber`, `isMaleNumber` som ger `true` om det är ett manligt respektive kvinnligt personnummer.
3. `areEqual` som jämför två Pnr-instanser och ser om de representerar samma personnummer.
4. **(VG-uppgift)** `isCorrect` som ger `true` om det är ett korrekt svenskt personnummer. Förutom att kontrollera att datumet är vettigt skall ni också kontrollera att personnumrets sista siffra (kontrollnumret) är korrekt. Mer information om detta hittar ni på Wikipedia.

Lägg gärna till fler metoder om ni tycker att något fattas. Lämpliga argument- och returtyper för de olika metoderna får ni bestämma själva.

Notera: Deluppgifterna 1-3 är obligatoriska. Deluppgift 4 (`isCorrect`) är en VG-uppgift.

Lektion 7 - Skapa egna klasser

I uppgifterna 3-6 nedan skall ni skriva egna klasser. Vi vill också att ni till varje klass (tex `MultiDisplay`) skriver ett testprogram (tex `MultiDisplayMain`) som innehåller en `main`-metod som visar hur klassens olika metoder kan användas.

- **Uppgift 3**
  Skapa en klass `MultiDisplay` som vid exekveringen av denna kod:

```
MultiDisplay md = new MultiDisplay();

md.setDisplayMessage("Hello World!");
md.setDisplayCount(3);
md.display();                           // ==> print-out

md.display("Goodbye cruel world!", 2);  // ==> print-out

System.out.println("Current Message: "+ md.getDisplayMessage());
```

  ger denna utskrift:

```
Hello World!
Hello World!
Hello World!
Goodbye cruel world!
Goodbye cruel world!
Current Message: Goodbye cruel world!
```

  Klassen `MultiDisplay` skall förstås klara av andra meddelanden och andra antal upprepningar.

- **Uppgift 4**
  Ladda hem och installera klassen `AlarmClock` . Skriv sedan ett program `AlarmMain` som använder `AlarmClock` för att
  1. Ställa väckarklockan till tiden 23:48
  2. Visa tiden
  3. Ställa in väckningstiden till 6:15
  4. Låta klockan gå 500 minuter
  5. Visa tiden igen

  Notera: Du får inte göra några förändringar i klassen `AlarmClock` förutom att kanske ändra på paketets namn.

- **Exercise 5**
  Skapa en klass `Radio` som när man exekverar den här koden:

```
System.out.println("Radio 1");
Radio r1 = new Radio();
System.out.println( r1.getSettings());  // Default settings

// Update Radio 1 settings
r1.turnOn();
r1.setVolume(3);
r1.channelUp();
r1.channelUp();
r1.channelUp();
System.out.println( r1.getSettings()); // New settings

r1.turnOff();
System.out.println( r1.getSettings());  // Reset default settings

System.out.println("\nRadio 2");
Radio r2 = new Radio();
r2.volumeUp();   // Radio off ==> No adjustment possible

r2.turnOn();
r2.volumeDown();   // volume = 0 ==> OK!
r2.volumeDown();   // volume < 0 ==> error and neglect
r2.setChannel(15); // out of range = ==> error and neglect
System.out.println( r2.getSettings());
```

ger följande utskrift:

```
Radio 1
Settings: On false, Channel 1, Volume 1
Settings: On true, Channel 4, Volume 3
Settings: On false, Channel 1, Volume 1

Radio 2
Radio off ==> No adjustment possible
New volume not within range!
New channel not within range!
Settings: On true, Channel 1, Volume 0
```

Notera

- Radions default-inställning (eng. Settings) är `on = false, channel = 1, volume = 1`.
- Volymen är alltid i intervallet [0,5]
- Kanalen är alltid i intervallet [1,10]
- Du kan inte ändra kanal eller volym när radion är avstängd.

Inställningar utanför de givna intervallen, eller då radion är avslagen, skall generera ett felmeddelande (och ignoreras).

Slutligen, klassen `Radio` skall (åtminstone) innehålla följande nio metoder:

```
getSettings()  // A string with current settings
turnOn(), turnOff()  // turnOff() ==> restore default settings
setVolume(int newVolume), volumeUp(), volumeDown() // up/down ==> +- 1 step
setChannel(int newChannel), channelUp(), channelDown() // up/down ==> +- 1 step
```

Tips: Försök inte att implementera alla "features" på en gång. Börja med en inställning (säg Volume) och försök få den att fungera utan några intervall. Lägg sedan till intervall, lägg sedan till den andra inställningen.

- **Uppgift 6**
  Skapa en klass `Point` som vid exekveringen av denna kod:

```
Point p1 = new Point();
Point p2 = new Point(3,4);

System.out.println(p1.toString());   // ==> (0,0)
System.out.println(p2.toString());   // ==> (3,4)

if (p1.isEqualTo(p2))              // False!
        System.out.println("The two points are equal");

double dist = p1.distanceTo(p2);
System.out.println("Point Distance: "+dist);

p2.move(5,-2);         // ==> (8,2)
p1.moveToXY(8,2);      // ==> (8,2)

if (p1.isEqualTo(p2))              // True!
        System.out.println("The two points are equal");
```

ger denna utskrift:

```
(0,0)
(3,4)
Point Distance: 5.0
The two points are equal
```

Klassen `Point` skall förstås hantera andra punkter med andra (x,y)-värden på ett vettigt sätt. Notera:

- Koordinaterna (x,y) är alltid heltal.
- Metoden `toString` ger en sträng med punktens aktuella koordinater lämplig för utskrift.
- Avståndet mellan punkter beräknas på samma sätt som i Uppgift 15, Lab 1.
- Två punkter är lika (equal) om de har samma koordinater.
- Metoden `move` flyttar punkten i x- och y-led.
- Metoden `moveToXY` ger punkten helt nya koordinater.

Lektion 8 - Mera klasser

I uppgifterna 7-9 nedan skall ni skriva egna klasser. Vi vill också att ni till varje klass (tex `Fraction`) skriver ett testprogram (tex `FractionMain`) som innehåller en `main`-metod som visar hur klassens olika metoder kan användas.

- **Uppgift 7**
  Skapa en klass `Fraction.java` som representerar ett bråktal på formen T/N där T (täljaren) och N (nämnaren) är heltal. Om nämnaren är noll skall ett felmeddelande lämnas. Det skall finnas följande medlemmar i klassen.
  - En konstruktor som skapar och initialiserar ett nytt bråktal.
  - Metoderna `getNumerator` och `getDenominator` som returnerar täljaren respektive nämnaren.
  - Metoden `isNegative` som ger `true` om det är ett negativt bråktal.
  - Metoderna `add, subtract, multiply, divide` som utför motsvarande bråktalsoperation på två bråk och som returnerar ett nytt bråktal. Bestäm själv ett lämpligt sätt att hantera de fall där någon av de inblandade bråktalen har noll i nämnaren.
  - `isEqualTo` som jämför två Fraction-instanser och ser om de representerar samma bråktal.
  - `toString` som returnerar en strängrepresentation av bråktalet på form T/N.

  Lägg gärna till fler metoder om ni tycker att något fattas. Lämpliga argument- och returtyper för de olika metoderna får ni bestämma själva.

  Överkurs för den matematikintresserade: Se till så alla bråktal alltid är maximalt förenklade. T ex, bråktalen 2/4 och 35/50 skall internt representeras som 1/2 och 7/10. D v s, den interna

representationen skall alltid vara de två minsta möjliga heltalen T och N som representerar samma rationella tal T/N. Här kan det vara nyttigt att titta på Euklides algoritm (se Wikipedia).

- **Uppgift 8**
  Skriv en klass `Card` som representerar ett kort i en vanlig kortlek med 52 olika kort. Varje kort har en *färg* (4 möjliga) och en *valör* (13 möjliga). Skriv sedan en klass `Deck` som från början innehåller 52 olika objekt av klassen `Card`. Klassen `Deck` skall ha metoder för att blanda kortleken, dela ut ett kort och upplysa om hur många kort som finns kvar. Notera, det skall bara gå att blanda kortleken när den har 52 kort.

  Skriv också ett program `DeckMain` som skapar en kortlek, delar ut ett par kort, och sedan visar hur många kort det finns kvar och vilka kort som delats ut.

  Ledtråd: Använd uppräkningstyper (eng. Enumeration Types).

- **Uppgift 9 (VG-Uppgift)**
  I denna uppgift skall ni använda klassen `Deck` från föregående uppgift.
  I patiensen 1-2-3 lägger man ut ett kort åt gången i en hög, samtidigt som man räknar 1,2,3,1,2,3,1,2,3 osv. Man förlorar patiensen så fort ett ess kommer då man säger "ett", en tvåa då man säger "två", eller en trea när man säger "tre". Som ni förstår är den väldigt svår att få ut, men hur svår?

  Skriv ett program Play123Main som lägger patiensen 1-2-3 10000 gånger och sedan beräknar sannolikheten (%) för att patiensen går ut. Programmet skall använda en metod `play123` som lägger patiensen 1 gång och returnerar `true` om den går ut.

Redovisning

Alla uppgifter skall lämnas in och vi är bara intresserade av era .java- filer (och VG-uppgifterna 2.4 och 9 är inte obligatoriska). Zippa därför ihop .java-filerna i katalogen DittLnuAnvändarNamn_lab3 (som finns inuti katalogen src) och lämna in dem mha Moodles redovisningssystem.