Inlämningsuppgift 2 - Datastrukturer

Avsikt

Avsikten med inlämningssuppgiften är att komplettera paketet *collections* med ett antal användbara klasser. Du ska också skriva ett mindre program vilket ger vägbeskrivning mellan två orter i skåne.

Resurser

I arkiv-filen ***DA353AVT14_P2.zip*** hittar du följande filer:

- Graph.java, Edge.java, GraphSearch.java, Places.java, Road.java, Position.java och MapView.java
- places.txt, roads.txt, skane.jpg (placera dem i M:\files)

Redovisning

Din lösning av uppgiften lämnas in via It's learning *senast kl 09.00 torsdagen den 10/3*. Du ska placera paketen *collections* och *p2* och katalogen *docs i* i en zip-fil. Enklast kopierar du *collections, p2* resp *docs* från Eclipse till en mapp på hårddisken. Sedan skapar du arkivfilen.

Vid redovisningen *fredagen den 11/3* kommer dina klasser i paketet *collections* att testköras. Kontrollera därför noga deras funktion innan inlämningen.

Zip-filen ska du ge namnet AAABBBP2.zip där AAA är de tre första bokstäverna i ditt efternamn och BBB är de tre första bokstäverna i ditt förnamn. Använd endast tecknen a-z när du namnger filen.

- Om Rolf Axelsson ska lämna in sina lösningar ska filen heta AxeRolP2.zip.
- Om Örjan Märla ska lämna in sina lösningar ska filen heta MarOrj.zipP2.
- Är ditt förnamn eller efternamn kortare än tre bokstäver så ta med de bokstäver som är i namnet: Janet Ek lämnar in filen EkJanP2.zip

Granskning

Ca *kl 13.00 den 10/3* kommer en kamrats lösning finnas i din inlämning på It's learning. Din uppgift är att granska kamratens lösningar på uppgifterna avseende:

- funktion – hur väl uppfyller lösningen kraven i uppgiften? Fungerar klasserna på avsett sätt?
- kan du tänka dig något alternativt sätt att lösa uppgiften?
- javadockommentarer – är klasserna kommenterade enligt instruktion? Och är kommentarena vettiga?
- genererad javadoc – är javadoc-dokument genererade? Är dokumenten vettiga? Fungerar länkar?

Resultatet av din granskning, 1-2 A4-sidor, ska du lämna in via It's learning senast 8.00 den 11/3.

Uppgift 1

Du ska placera följande interface i paketet *collections*:

- Queue<T>
- SearchTree<K,V>
- Map<K,V>

Du ska placera följande klasser i paketet *collections*:

- Minst en implementering av Queue<T>:
  - LinkedQueue<T> + QueueException
  - ArrayQueue<T> + QueueException
- PriorityQueue<T>. Välj en implementering. Tänk på att *laboration11.PriorityQueue* använder klassen *ArrayHeap*
- Minst en implementering av *SearchTree<K,V>*:
  - *BinarySearchTree<K,V> + BSTNode<K,V>*
  - *AVLTree<K,V> + AVLNode<K,V>*

  I båda fallen måste du avkommentera *showNode*-metoden alternativt ta med några klasser till.
- *HashtableOH<K,V> + Entry(K,V)*

Börja med att kopiera ovanstående interface till *collections*. Kopiera sedan en klass i taget. Ta bort alla importer från egna paket (kan bli sådana vid kopiering). Alla klasser och interface som används ska finnas i *collections*. Undantag är:

- Interfacen *Comparator*, *Comparable*, *Iterator* och *Iterable*.
- Vissa Exception-klasser vilka är standardklasser i java.

Testkör alltid objektsamlingen du just kopierat in innan du fortsätter med nästa.

Det är naturligtvis tillåtet att lägga till fler klasser i *collections*.

Uppgift 2

Javadoc-kommentera samtliga klasser i paketet *collections* (kommentera på svenska eller engelska, använd samma språk i hela klassen). Följande kommentarer ska finnas för varje klass:

Ovanför klassdeklarationen:

Dokumentation över vad klassen kan användas till.
Dokumentation över författare av klassen (du själv, använd @author-taggen)

```
/**
 * Resizable-array implementation of the List interface. An ordered sequence where
 * the user has precise control over where in the list each element is inserted.
The
 * user can access elements by their integer index (position in the list), and
 * search for elements in the list.
 *
 * @author Rolf Axelsson
 */
public class ArrayList<E> implements List<E> {
 :
}
```

Ovanför varje public-deklarerad konstruktor:

Dokumentation över vad som konstrueras och över parametrar. En *@param*-tag för varje parameter.

```
/**
 * Constructs an empty list with the specified initial capacity.
 *
 * @param initialCapacity the initial capacity of the list
 */
public ArrayList(int initialCapacity) {
```

```
  :
}
```

Ovanför varje public-deklarerad metod:

Dokumentation över vad metoden kan användas till. En *@param*-tag för varje parameter. *@return*-tagg om metoden returnerar ett värde (ej void).

```
/**
 * Returns the index of the first occurrence of the specified element in this list,
 * or -1 if this list does not contain the element. Search begins at position
 * startIndex in the list. equals is used for comparing elements.
 *
 * @param begins at position startIndex in the list
 * @param element element to search for
 * @return the index of the first occurrence of the specified element in this list,
 * or -1 if this list does not contain the element
 */
public int indexOf(E element) {
  :
}
```

Uppgift 3

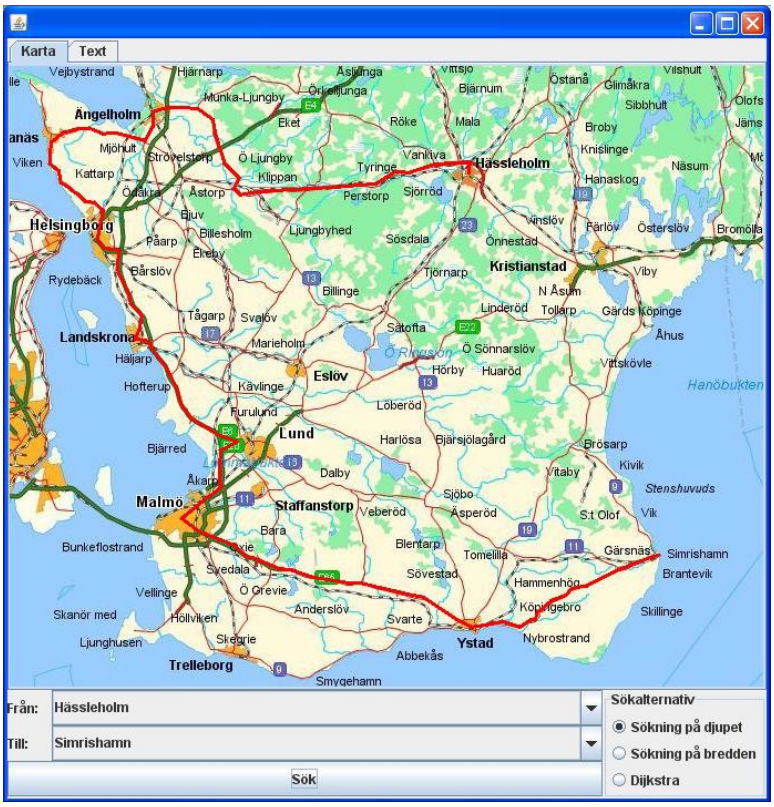
Generera javadoc-dokumentation med hjälp av eclipse (dokumenten hamnar i *doc*-katalogen).

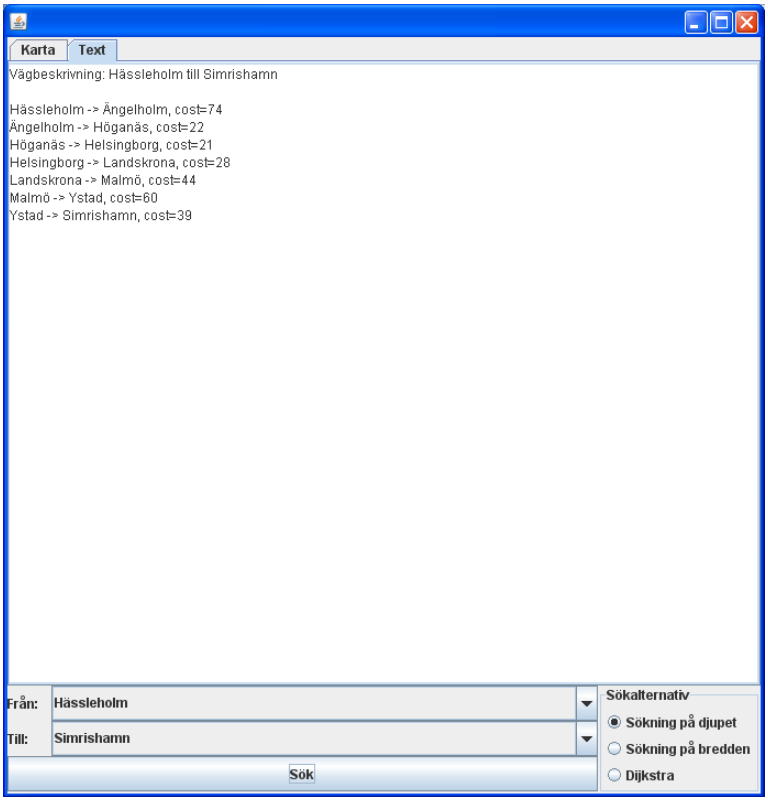
Välj *Project – Generate javadoc…* Se till att endast paketet *collections* är markerat och klicka på *finish*.

Uppgift 4

I uppgift 4 ska du skriva ett program vilket visar vägen mellan två orter på en karta. Till din hjälp har ska du bl.a. klasserna Graph, Edge, GraphSearch, Places, Road, Position och MapView i paketet *p2*. Användaren ska kunna begära en vägbeskrivning mellan två orter. Och resultatet beror på vilken sökalgoritm som användaren anget att programmet ska använda för att generera vägbeskrivningen.

I figuren nedan har användaren valt att få en vägbeskrivning från Hässleholm till Simrishamn, och sökmetoden är sökning på djupet. På nästa sida ser du hur vägbeskrivningen kan se ut i textform.





Om”Sökning på bredden” eller ”Dijkstra” är markerat så visas (oftast) ett annat sökreultat.

Krav på systemet

Klassen *MainP2* ska starta exekveringen av programmet. Du får ej ändra i klassen.

```
package p2;
import javax.swing.SwingUtilities;

public class MainP2 {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                Position mapLeftUp = new Position(12.527, 56.346);
                Position mapRightDown = new Position(14.596, 55.324);
                new P2Controller("files/skane.jpg", mapLeftUp,
mapRightDown, "files/places.txt","files/roads.txt");
            }
        });
    }
}
```

Klassen / klasserna som hanterar det grafiska användargränssnittet ska inte innehålla någon logik. Det innebär att händelser ska meddelas *P2Controller* och *P2Controller* styr vad som ska visas i det grafiska användargränssnittet.

I exemplet ovan används JComboBox-komponenter vid val av orter. På så sätt kan användaren aldrig ange en ort som inte finns i grafen. Men det går även bra med vanliga inmatningsfönster för att ange orterna.

I exemplet ovan används JTabbedPane för att användaren ska kunna skifta mellan kartbeskrivning och textbeskrivning. Det gå bra att visa både karta och textbeskrivning samtidigt (kanske t.o.m. en fördel?)

Placera klasserna du själv skriver i paketet *p2*.