Laboration 5 – länkad lista

Hämta filen **L5_kod.zip** från kurssidan och packa upp innehållet i **paketet laboration7**. Föreläsningsfilerna i **F6_kod.zip** ska vara i paketet **f7**.

Grundläggande

Uppgift 1

Uppgift 1 handlar om att komplettera ett antal metoder i klassen ***ArrayList<E>*** i paketet *f7*. Efter att du färdigställt en metod så ska du kontrollera att den fungerar som tänkt.

a. Komplettera metoden ***grow*** med kod så att fler objekt kan lagras i arrayen elements. Gör så här:
   - Skapa en array med dubbelt så stor kapacitet som elements, E[] temp = … // se konstruktor
   - Kopiera elementen från *elements* till *temp*
   - Tilldela *elements* referensen till *temp*.

   Testa din lösning genom att skapa en *ArrayList* med liten initial kapacitet och kontrollera att den växer som tänkt.

b. Komplettera metoderna ***addFirst(E)*** och ***addLast(E)***.

c. Komplettera metoden ***remove(int index)***. Följande måste du tänka på:
   Är *index* korrekt, annars kasta *IndexOutOfBoundsException*.
   Lagra elementet (objektet) i angiven position (*index*) i en lokal variabel, E element = …
   Minska *size* med 1.
   Flytta alla element efter det som ska tas bort ett steg mot listans start.
   returnera *element*.

d. Komplettera metoderna ***removeFirst()*** och ***removeLast()***.

e. Komplettera metoden get(int index). Tänk på följande:
   Är *index* korrekt, annars kasta *IndexOutOfBoundsException*.
   Returnera elementet i vald position

f. Komplettera metoden ***set(int index, E element)***. Tänk på följande:
   Är *index* korrekt, annars kasta *IndexOutOfBoundsException*.
   Lagra elementet i angiven position (*index*) i en lokal variabel, E prevElement = …
   Lagra *element* i arrayen
   returnera *prevElement*

g. Komplettera metoden ***clear()***. Metoden ska tömma ArrayList-objektet på element. Det är bra om samtliga element i *elements* sätts till *null* vid tömningen. Glöm inte nollställa *size*.

h. Komplettera metoden ***indexOf(int startIndex, E element)***. Tänk på följande:
   Är *startIndex* korrekt, annars kasta *IndexOutOfBoundsException*.
   Börja sökningen på elementet i position startIndex och håll på tills du funnit element som söks eller det inte finns fler element i arrayen. Vid sökningen ska du jämföra elementen med *equals*-metoden. Sökningen går för övrigt till som en vanlig linjär sökning.

i. Komplettera metoden ***indexOf(E element)***.

j. Komplettera metoden ***size()***.

Uppgift 2

I Uppgift 2 ska du komplettera metoder i klassen ***Laboration7a*** med kod. Var noga med att skissa din lösning på papper innan du skriver raderna med kod. Uppgifterna kräver att klassen *ObjectNode* finns i paketet *f7*.

a. Rita upp den länkade listan som skapas i början av metoden ***exercise2a*** på ett papper. Varje *ObjectNode*-objekt ska utgöras av en tvådelad rektangel och pilar ska visa hur objekten refererar till varandra.
Kompletter sedan metoden med kod som lägger till värdet 7 i position 4 i listan. Om du anropar metoden *exercise1* efter du kompletterat med kod ska du få utskriften

```
[ 23, 17, 20, -5, -9, 11, 9 ]
[ 23, 17, 20, -5, 7, -9, 11, 9 ]
```

b. Komplettera metoden ***exercise2b*** med kod som skriver ut listan på formen:
```
[ 23 17 20 -5 -9 11 9 ]
```

c. Komplettera metoden ***exercise2c*** så att följande utskrifter sker då du anropar metoden:

```
[ 23, 17, 20, -5, -9, 11, 9 ]
[ 23, 17, -14, 4, 20, -5, -9, 11, 9 ]
```

d. Komplettera metoden ***exercise2d*** så att följande utskrifter sker då du anropar metoden:

```
[ 23, 17, 20, -5, -9, 11, 9 ]
[ 23, 17, -14, 20, 4, -5, -9, 11, 9 ]
```

e. Komplettera metoden ***exercise2e*** med kod så att elementet i position 3 tas bort från listan. När metoden anropas ska du få utskrifterna

```
[ 23, 17, 20, -5, -9, 11, 9 ]
[ 23, 17, 20, -9, 11, 9 ]
```

f. Komplettera metoden ***exercise2f*** med kod så att elementet i position 0 tas bort från listan. När metoden anropas ska du få utskrifterna

```
[ 23, 17, 20, -5, -9, 11, 9 ]
[ 17, 20, -5, -9, 11, 9 ]
```

g. Komplettera metoden ***exercise2g*** med kod så att elementen i position 0 och 1 tas bort från listan. När metoden anropas ska du få utskrifterna

```
[ 23, 17, 20, -5, -9, 11, 9 ]
[ 20, -5, -9, 11, 9 ]
```

h. Komplettera metoden ***exercise2h*** med kod så att elementen i position 0 och 1 tas bort från listan. När metoden anropas ska du få utskrifterna

```
[ 23, 17, 20, -5, -9, 11, 9 ]
[ 20, -5, -9, 11, 9 ]
```

i. Komplettera metoden ***exercise2i*** med kod så att elementen i position 1 och 3 tas bort från listan. När metoden anropas ska du få utskrifterna

```
[ 23, 17, 20, -5, -9, 11, 9 ]
[ 23, 20, -9, 11, 9 ]
```

Uppgift 3

I var och en av deluppgifterna ska du implementera en metod i en klass. I samtliga uppgifter implementerar klassen en länkad lista som lagra objekt. Samtliga klasser är alltså identiska men olika metoder saknas. Din uppgift är att komplettera med den saknade metoden utan att titta på hur metoden är implementerad i de andra uppgifterna.

När du löser en uppgift ska du:
1. skissa din lösning på papper – hur du steg för steg ska gå till väga
2. implementera din lösning.
3. jämföra din lösning med hur metoden är skriven i någon av de andra uppgifterna.

a. Klassen **ObjectListSize** innehåller allt som finns i klassen **ObjectList** utom ett antal rader med kod i metoden ***size***. Metoden *add* är lite modifierad men det behöver du inte bry dig om.
Din uppgift är att komplettera metoden *size* med kod. Jämför sedan din lösning med *size*-metoden i klassen ObjectList (paketet *f7*).

Om du exekverar main-metoden får du körrresultatet:
```
[ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ]
Size = 0
```

När du lagt till kod i size-metoden ska körresultatet vara:
```
[ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ]
Size = 10
```

b. Klassen **ObjectListGet** saknar ett antal rader med kod i metoden ***get***. Kompletta *get*-metoden med kod så du får önskvärt körresultat:
```
[ 0, 1, 2, 3, 4 ]
java.lang.IndexOutOfBoundsException: size=5, index=-1
Element 0 = 0
Element 1 = 1
Element 2 = 2
Element 3 = 3
Element 4 = 4
java.lang.IndexOutOfBoundsException: size=5, index=5
```

c. Klassen **ObjectListAdd** saknar ett antal rader med kod i metoden ***add***. Kompletta *add*-metoden med kod så du får önskvärt körresultat. När du är färdig med din lösning ska körresultatet bli:
```
[ 0, 0, 1, 1, 2, 2, 3, 3, 4, 4 ]
```

d. Klassen **ObjectListRemove** saknar ett antal rader med kod i metoden ***remove***. Kompletta *remove*-metoden med kod så du får önskvärt körresultat. När du är färdig med din lösning ska körresultatet vara:
```
[ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ]
4
3
2
1
0
[ 5, 6, 7, 8, 9 ]
```

Fördjupande

Uppgift 4

Komplettera klassen **ObjectList** (i paketet *f7*) med följande metoder:

a. **addFirst(int data)** vilken ska placera värdet data först i listan

b. **addLast(int data)** vilken ska placera värdet data sist i listan

c. **removeFirst()** vilken ska ta bort det första elementet ur listan och returnera värdet i det borttagna elementet.

d. **removeLast()** vilken ska ta bort det sista elementet ur listan och returnera värdet i det borttagna elementet.

e. **set(int index, int data)** vilken ska ändra värdet i elementet i position index.

Testa metoderna så deras funktion är korrekt.

Uppgift 5

Uppgift 5 handlar om att komplettera metoder i klassen ***LinkedList<E>*** (paketet *f7*) med kod, så att metoder i ***Laboration7b*** får avsett körresultat.

a. Metoden ***exercise5a*** i *Laboration7b*
Innan du kompletterat ***add(int,E)*** får du körresultatet: `[]`
Efter komplettering ska du få körresultatet: `[ 0; 1; 2; 3; 4; 5; 6; 7; 8 ]`

b. Metoden ***exercise5b*** i *Laboration7b*
Innan kompletteringar av övriga ***add***-metoder får du körresultatet: []
Efter kompletteringar får du körresultatet: `[ 6; 5; 4; 0; 1; 2; 8; 9; 10 ]`

c. Metoden ***exercise5c*** i *Laboration7b*
Innan kompletteringar av ***remove***-metoder får du körresultatet:
```
Borttagen: null
Borttagen: null
Borttagen: null
[ 0; 1; 2; 3; 4; 5; 6; 7; 8; 9 ]
```
Efter kompletteringar får du körresultatet:
```
Borttagen: 0
Borttagen: 9
Borttagen: 1
[ 2; 3; 4; 5; 6; 7; 8 ]
```

d. Metoden ***exercise5d*** i *Laboration7b*
Innan komplettering av ***set***-metoden får du körresultatet: `[ 0; 1; 2; 3; 4; 5; 6; 7 ]`
Efterkomplettering får du körresultatet: `[ 10; 1; 2; 13; 4; 5; 16; 7 ]`

e. Metoden ***exercise5e*** i *Laboration7b*
Innan kompletteringar av ***indexOf***-metoder får du körresultatet: `-1, -1, -1, -1, -1`
Efter komplettering av ***indexOf(int,E)*** får du körresultatet: `5, -1, -1, -1, -1`
Efter komplettering av ***indexOf(E)*** får du körresultatet: `5, -1, -1, -1, 5`

Extra

Uppgift 6

Det återstår några metoder att implementera i ***LinkedList<E>***-klassen.

a. Metoden ***exercise6a*** i *Laboration7b*
Innan komplettering av ***clear***-metoden får du körresultatet `[]`. Det är ju bra. Ett problem är att alla objektreferenser finns kvar i *ListNode*-objekten. Varvid det blir svårare för garbage-collectorn att återanvända utrymmet som objekten använt.
Se till att samtliga *data*-referenser är *null* och samtliga *next*-referenser är *null* i den gamla listan innan *list* tilldelas värdet *null*.
Ledning:
Skapa ett ArrayStack<E>-objekt (*f6*).
Placera samtliga ListNode-objekt i listan till stacken (*push*)
Medan det är fler element på stacken
  hämta ett element (*pop*)
  tilldela *data*-referensen värdet *null*
  tilldela *next*-Referensen värdet *null*

Ett alternativ till ovanstående lösning är en *rekursiv* metod:

```
private clear(ListNode<e> node) {
    om(node.getNext()!=null)
        clear(node.getNext());
    sätt referenserna till null
}
```

b. Metoden ***exercise6b*** i *Laboration7b*
Innan komplettering av den inre klassen ***Iter*** returneras aldrig några *data*-referenser av *next*- metoden och *hasNext*-metoden returnerar alltid *true*. Komplettera klassen så att samtliga data-objekt i listan returneras av next-metoden och att hasNext returnerar true så länge det finns fler data-objekt att returnera via next-metoden.
Före komplettering blir utskriften så här:

```
[ 0; 1; 2; 3; 4; 5; 6; 7; 8; 9 ]
```

Efter komplettering ska den bli så här:

```
[ 0; 1; 2; 3; 4; 5; 6; 7; 8; 9 ]
0 1 2 3 4 5 6 7 8 9
```

c. För att den förenklade for-loopen ska kunna användas på en objektsamling så måste samlingen implementera interfacet Iterable

```
public interface Iterable<E> {
    public Iterator<E> iterator();
}
```

Eftersom klasserna *ArrayList<E>* och *LinkedList<E>* båda innehåller metoden *iterator()* så behöver du endast göra ett tillägg i klassdeklarationen för att den förenklade for-loopen ska kunna användas.

```
public class ArrayList<E> implements List<E>, Iterable<E> {
    :
}
```

Gör detta tillägg och testa sedan funktionen.

Lösningar

Uppgift 1

Inga lösningar

Uppgift 2a

```
ObjectNode newNode = new ObjectNode( 7, pos4 );
pos3.setNext( newNode );
```

eller

```
pos3.setNext( new ObjectNode( 7, pos4 ) );
```

Uppgift 2b

```
System.out.print( "[ " );
ObjectNode node = list;
while( node != null ) {
    System.out.print( node.getData() + " " );
    node = node.getNext();
}
System.out.print( "]" );
```

Uppgift 2c

```
ObjectNode new2 = new ObjectNode( 4, pos2 );
ObjectNode new1 = new ObjectNode( -14, new2 );
pos1.setNext(new1);
```

eller

```
pos1.setNext( new ObjectNode( -14, new ObjectNode( 4, pos2 ) ) );
```

Uppgift 2d

```
pos2.setNext( new ObjectNode( 4, pos3 ) );
pos1.setNext( new ObjectNode( -14, pos2 ) );
```

Uppgift 2e

```
pos2.setNext(pos4);
```

Uppgift 2f

```
list = list.getNext();
```

Uppgift 2g

```
list = pos2;
```

Uppgift 2h

```
list = list.getNext().getNext();
```

Uppgift 2i

```
list.setNext( pos2 );
pos2.setNext( pos4 );
```

Uppgift 3

Inga lösningar

Uppgift 4

```
public void addFirst(int data) {
    add(0, data);
}

public void addLast(int data) {
    add(size(), data);
}

public int removeFirst() {
    return remove(0);
}

public int removeLast() {
    return remove(size()-1);
}

public void set( int index, Object data ) {
    if( ( index < 0 ) || ( index >= size() ) )
        throw new IndexOutOfBoundsException( "size=" + size() + ", index="
+ index );

    ObjectNode node = locate( index );
    Object prevData = node.getData();
    node.setData( data );
    return prevData;
}
```

Uppgift 5

Inga lösningar