Objektorienterad Programmering (TDDC77)

Föreläsning VII: felhantering(kont.), abstrakta datatyper

Ahmed Rezine

IDA, Linköpings Universitet

Höstermin 2016

Outline

Felhantering

ADT

Köer (Queues)

Felhantering

- Exception in thread "**main**" **java.lang.NumberFormatException**: 23.5
- En del metoder och operationer kan kasta ut fel
- I API:et står vilka fel som kastas för en viss metod
- Integer.parseInt(String) kastar ett NumberFormatException
- Alla fel kan fångas och hanteras med **try/catch**-konstruktionen

Läs in heltal

```
int number = 0;
print ("Mata in ett tal ");
number = Integer.parseInt(in.nextLine());
println("Du matade in talet: " + number);
```

- Fungerar för det mesta, men vad händer om man matar in "2.5" eller "e" ?

Läs in heltal med felhantering

```
boolean valueOK = true;
int number = 0;
do{
    valueOK = true;
    print ("Mata in ett tal ");
    Scanner in = new Scanner (System.in);
    String line = in.nextLine();
    try{
     number = Integer.parseInt(line);
    } catch (NumberFormatException e){
     System.out.println(line + " är inget tal");
     valueOK = false;
    }
} while (!value OK);
System.out.println("Du matade in talet: " + number);
```

Hantera felet

- **java.lang.ArrayIndexOutOfBoundsException**

```
int[] list = {45, 34, 67, 98};
while(true){
    print("Ange ett index: ");
    int index = Integer.parseInt(in.nextLine());
    print("Resultat: " + list[index]);
}
```

Hantera felet

- Kan man undvika att ett fel inträffar från första början så är det bättre än try/catch
- Vilket/vilka fel kan man undvika i arrayindexeringsexemplet?

Scanner och Slingor: exempel

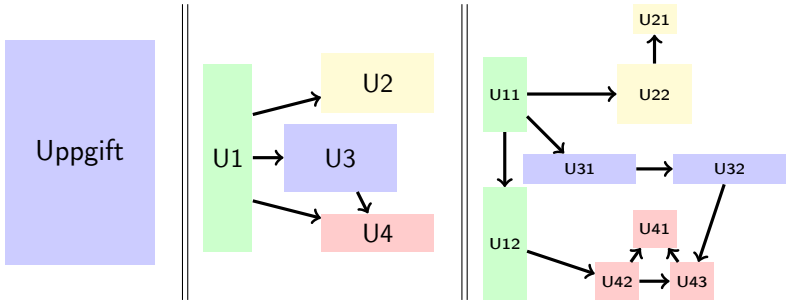
```
import java.util.Scanner;
import java.io.*;

public class HittaKonto{

    static public void main(String[] args) throws IOException{
     Scanner namnScan, filScan, radScan;
     String rad, namn;

     namnScan = new Scanner(System.in);
     System.out.print("Efternamn ? ");
     namn = namnScan.next();

     filScan = new Scanner(new File("it-konton-2016.txt"));

     while(filScan.hasNext()){
         rad = filScan.nextLine();
         radScan = new Scanner(rad);
         String radKod = radScan.next();
         while(radScan.hasNext()){
          String radNamn = radScan.next();
          if(radNamn.equalsIgnoreCase(namn))
              System.out.println("Matchar: " + rad);
         }
     }
    }
}
```

Outline

Felhantering

ADT

Köer (Queues)

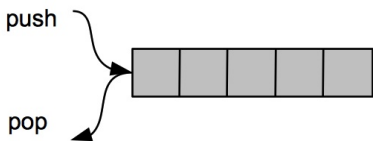


Abstrakta datatyper

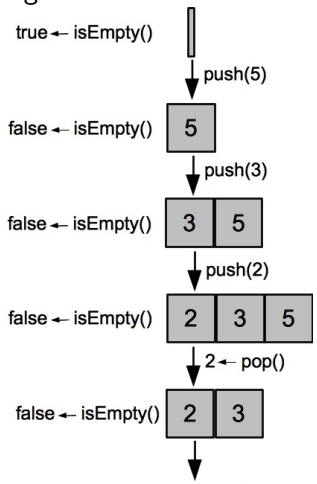
- Större abstraktioner som byggs av de kända byggstenarna
- Har ofta speciella funktioner för att bearbeta datatypen
- Idag kommer vi att titta på två exempel: köer (queues) och stapeln (stacks)
- LinCalc ...

Vad är en stack?

På svenska ungefär "stapel". Exempelvis en stapel av tallrikar. Last in first out (Lifo).



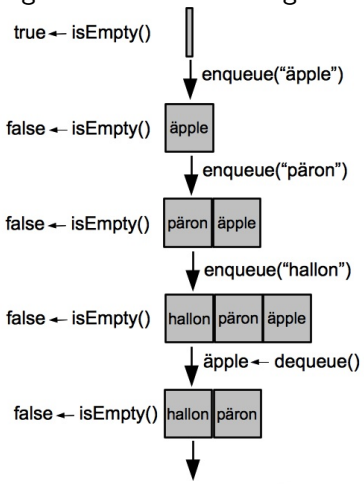
E.g. stack med heltal



Vad är en kö (queue)?

First in first out (Fifo)

E.g. kö med stränger



Fördelarna med ADT

- Abstrahera från det konkreta implementationen
- Använda metoder för att manipulera de abstrakta datatyper, i stället för att direkt komma åt de interna variablerna
- Kan ändra hur datatypen är implementerad utan att behöva ändra den kod som använder de abstrakta datatyper

Outline

Felhantering

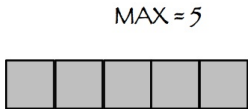
ADT

Köer (Queues)

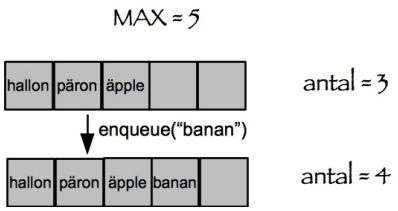
Implementera en kö med en array

- Använd en array (**MAX** storlek) för att bygga en "kö" av strängar
- Metoden **void enqueue(String s)** lägger till **s** sist i kön
- Metoden **boolean isEmpty()** returnerar sant omm kön är tom
- Metoden **String dequeue()** tar bort och returnerar den första strängen i kön

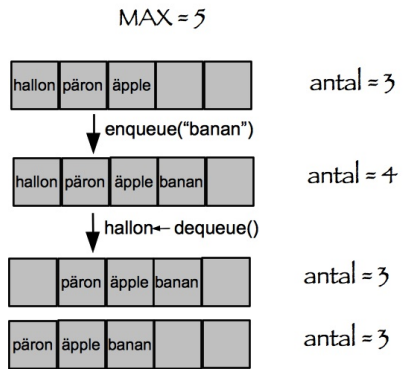
Implementera en kö med en array



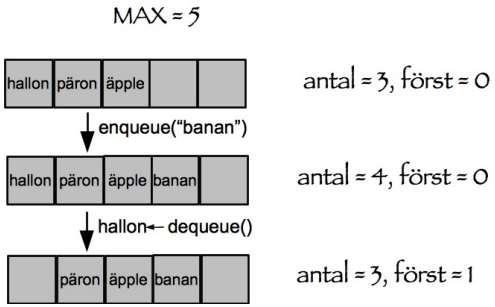
Implementera en kö med en array



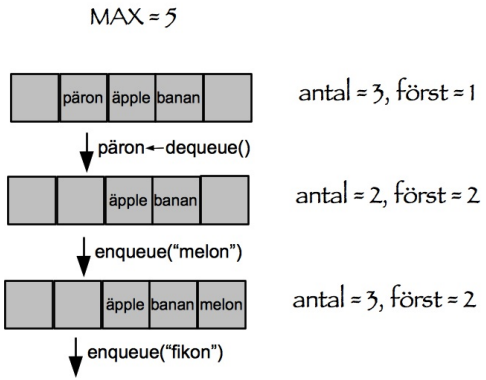
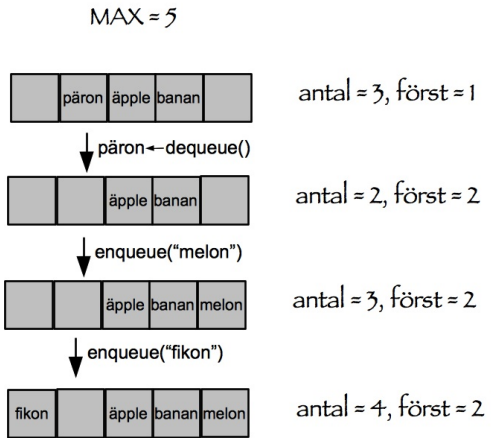
Implementera en kö med en array



Implementera en kö med en array



Implementera en kö med en array



Implementera en kö med en array