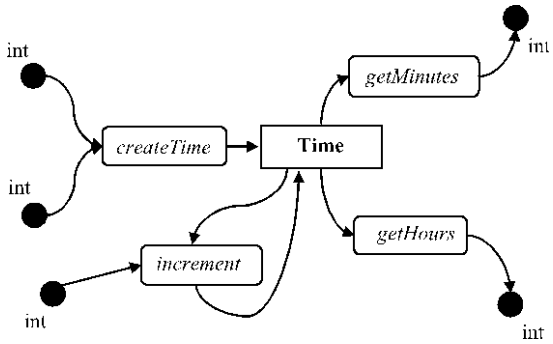
Abstraktní datové typy: zásobník

doc. Ing. Miroslav Beneš, Ph.D.
katedra informatiky FEI VŠB-TUO
A-1007 / 597 324 213
http://www.cs.vsb.cz/benes
Miroslav.Benes@vsb.cz

Abstraktní datové typy

- omezené rozhraní datového typu pomocí specifikované sady operací:
  - Time: getHours(), getMinutes(), increment()
- umožňují oddělení *rozhraní* od *implementace*
  - různá efektivita operací
  - různé vývojové verze téže implementace
  - implementace poskytnuté různými autory

Diagram signatury abstraktního datového typu



ADT a Java

- Rozhraní datového typu
  - pomocí konstrukce **`interface`**
  - není konstruktor, žádná data – jen operace bez implementace

```
public interface Time {
   int getHours();
   int getMinutes();
   int increment();
}
```

ADT a Java

- Implementace datového typu
  - konstruktory
  - instanční proměnné
  - implementace operací (musí být public!)

```
public class TimeImpl implements Time {
   public TimeImpl(int hrs, int mins) {
      m = hrs * 60 + mins;
   }
   public int getHours() { return m / 60; }
   public int getMinutes() { return m % 60; }
   public int increment() { m++; }
   private int mins;
}
```


Základní ADT

- zásobník (stack)
- fronta (queue)
- kolekce
  - množina (set)
  - multimnožina (bag)
  - seznam (list)
- zobrazení (map)
- strom (tree)

```
import java.util.*;
```


Zásobník

- Struktura LIFO (Last In – First Out)
- Operace:
  - push – vložení na vrchol zásobníku
  - pop – odstranění položky z vrcholu
  - top – získání hodnoty na vrcholu
  - empty – test, zda je zásobník prázdný

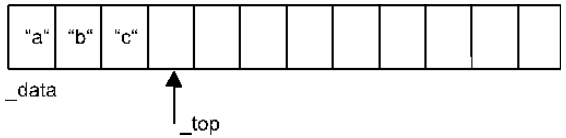
Zásobník - rozhraní

```
public interface Stack {
   void push(Object o);
   Object pop();
   Object top();
   boolean empty();
}
```

Zásobník - implementace

- Statická datová struktura
  - pole objektů s předem zadanou velikostí
  - při překročení kapacity se generuje výjimka
- Dynamická datová struktura
  - pole objektů s proměnnou velikostí
  - vázaná datová struktura

Zásobník ve statickém poli



- `Object[ ] _data = new Object[max];`
- `int _top = 0;`
- `0 <= _top <= _data.length (=max)`

Zásobník ve statickém poli

```
public class ArrayStack implements Stack
{
   public ArrayStack(int size) {
      assert size > 0;
      _data = new Object[size];
   }

   ...

   protected Object[]   _data;
   protected int        _top;
}
```


(Příkaz assert)

- **assert** podmínka;
  assert podmínka **:** „Popis chyby”;

```
if( !podmínka )
   throw new AssertionError(popis);
```

- **Překlad**: javac –source 1.4 Priklad.java
- **Spuštění**: java –ea Priklad

Zásobník ve statickém poli

```
   public void push(Object obj) {
      assert _top < _data.length : "Preteceni";
      _data[_top++] = obj;
   }

   public Object pop() {
      assert _top > 0 : "Zasobnik je prazdny";
      return _data[--_top];
   }

   public Object top() {
      assert _top > 0 : "Zasobnik je prazdny";
      return _data[_top-1];
   }

   public boolean empty() {
      return _top == 0;
   }
```


Zásobník ve statickém poli

```
class TestZasobniku {
  public static void main(String args[])
  {
    Stack z = new ArrayStack(4);
    System.out.println(z.empty());
    z.push("a");     z.push("b");
    z.push("c");     z.push("d");
    while( !z.empty() ) {
      System.out.print(z.pop()+" ");
    }
    System.out.println();
  }
}
```


Příklad – postfixový kalkulátor

```
class PostfixovyKalkulator {
  public void cislo(int x) {
    zasobnik.push(new Integer(x));
  }
  public void soucet() {
    int y = popInt();
    int x = popInt();
    zasobnik.push(new Integer(x+y));
  }
  public int vysledek() {
    return popInt();
  }
  private int popInt() {
    return ((Object)zasobnik.pop()).intValue();
  }
  private ArrayStack zasobnik = new ArrayStack(10);
}
```


Příklad – postfixový kalkulátor

```
PostfixovyKalkulator kalk =
   new PostfixovyKalkulator();

kalk.cislo(2);        // (2 + 7) + 4
kalk.cislo(7);
kalk.soucet();
kalk.cislo(4);
kalk.soucet();
System.out.println("Vysledek je " +
   kalk.vysledek());
```

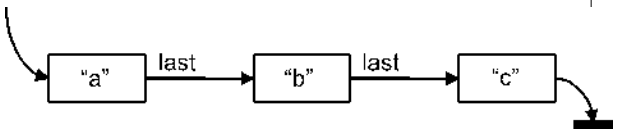
Zásobník v dynamickém poli

```
public class DArrayStack extends ArrayStack
{
  public DArrayStack(int size, int delta) {
    super(size);
    assert delta > 0;
    _delta = delta;
  }

  ...

  protected int _delta;
}
```


Zásobník v dynamickém poli

```
public void push(Object obj)
{
  if( _top == _data.length ) {
    Object[] nove =
                new Object[_top + _delta];
    for(int i = 0; i < _top; i++)
      nove[i] = _data[i];
    _data = nove;
  }
  super.push(obj);
}
```


Zásobník jako vázaná struktura



- Každý prvek nese hodnotu a odkaz na předchozí položku zásobníku.
- Položka na dně zásobníku nenese žádný odkaz (hodnota **null**)

Zásobník jako vázaná struktura

```
class StackElem {
   Object data;     // hodnota
   StackElem last;  // předcházející prvek

   StackElem(Object data, StackElem last)
   {
      this.data = data;
      this.last = last;
   }
}
```

Zásobník jako vázaná struktura

```
public class LinkedStack implements Stack
{
  public void push(Object obj) {
    _top = new StackElem(obj, _top);
  }

  public Object pop() {
    assert _top != null;
    Object topObj = _top.data;
    _top = _top.last;
    return topObj;
  }
```


Zásobník jako vázaná struktura

```
  public Object top() {
    assert _top != null;
    return _top.data;
  }

  public boolean empty() {
    return _top == null;
  }

  protected StackElem _top = null;
}
```


Aplikace zásobníku

- **Volání podprogramů**
  - ukládání aktivačních záznamů funkce na zásobník – návratová adresa, lokální proměnné a argumenty, mezivýsledky
  - umožňuje rekurzi
- **Vyhodnocování programu**
  - virtuální zásobníkový procesor
  - JVM – Java Virtual Machine (Sun)
  - CLR – Common Language Runtime (MS .NET)

Řešení úloh ze cvičení

```
public class Retezec
{
    public Retezec(int maxDelka) {
        data = new char[maxDelka];
    }

    public int length() { return delka; }

    private char[] data;
    private int delka;
}
```

Řešení úloh ze cvičení

```
public char charAt(int index) {
  if( index < 0 || index >= delka )
    throw new IndexOutOfBoundsException();
  return data[index];
}

public void setCharAt(int index, char ch) {
  if( index < 0 || index >= delka )
    throw new IndexOutOfBoundsException();
  data[index] = ch;
}
```


Řešení úloh ze cvičení

```
public void append(char ch) {
  if( delka == data.length )
    throw new IndexOutOfBoundsException();
  data[delka++] = ch;
}
public int indexOf(char ch) {
  for(int i = 0; i < delka; i++) {
    if( data[i] == ch ) return i;
  }
  return -1;
}
```


Řešení úloh ze cvičení

```
public String toString() {
  return new String(data, 0, delka);
}
public boolean equals(Object obj) {
  Retezec str = (Retezec)obj;
  if( delka != str.length() ) return false;
  for( int i = 0; i < delka; i++ ) {
    if( data[i] != str.charAt(i) )
      return false;
  }
  return true;
}
```