# Introductory Computing: The Design Discipline

**Viera Krňanová Proulx**

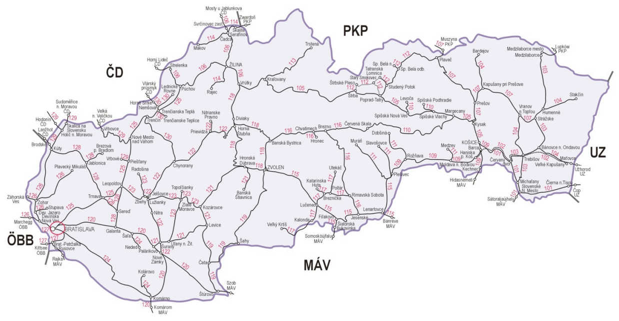
**Northeastern University, Boston, MA, USA**
**vkp@ccs.neu.edu**

## Informatics: The Science of Design

- Introduction: Design in the context of information systems
- Didactics: algebra and programming
- Information and data: the connection
- Design of abstractions = libraries
- Libraries for the beginners

## Informatics: The Science of Design

Concrete complex systems

- Automobiles
- Skyscrapers
- Slovak railroad system 

## Informatics: The Science of Design

Complex web pages

Systems for the weather forecasts

Medical information systems

Wikipedia

Computer games

Social computing

## Program by Design: Design-Based Introductory Curriculum

What are our goals?

- Computers are mostly about managing **data**
- **Data** comes in different forms
- **Data** represents information
- **Programs** transform data to yield new information
- **Abstractions** allow us to reuse programs, ideas

## Program by Design: Design-Based Introductory Curriculum

Simple language context

Systematic didactics for program design

Connection between information and data

Test-first design with appropriate support

Computer games - via libraries - focus on the model

Managing complexity through abstractions

## Program by Design: Design-Based Introductory Curriculum

**Bootstrap**

grades 6 - 8, very detailed curriculum

software support

**TeachScheme!**

secondary schools, introductory college

textbook, DrRacket student languages, libraries

**ReachJava**

second semester, class-based, object-oriented

libraries for game design, testing

textbook (draft), materials

## Didactics: Algebra and Programing

- Is the sum of three digits divisible by seven?
- Compute the location of a ball kicked in the given angle and with the given velocity
- Where is the given letter in the given String?

Functions with the given (input) data and output data

## Didactics: Algebra and Programing

Functions with the given (input) data and output data

**Didactics of the design of a function (a procedure)**

1. Think what are the inputs and outputs

2. Write down the purpose statement and the header (contract)

3. Make examples of use with expected outcomes

4. Inventory: make a list of all data parts and functions/methods/procedues that you can use

5. Design the body of the function/procedure

6. Use the examples from step three as test cases

## Didactics: Algebra and Programing 1

- **Is the sum of three digits divisible by seven?**
    - inputs: three digits output: boolean
- **Compute the location of a ball kicked in the given angle and with the given velocity**
    - inputs: angle, velocity, (gravitation) output: position (x,y)
- **Where is the given letter in the given String?**
    - inputs: character, String output: int

Analyze the problem, the types of inputs, outputs

## Didactics: Algebra and Programing 2

- **Is the sum of three digits divisible by seven?**
  - `// is the sum of three digits divisible by seven?`
  - `boolean divisibleBy7(int d1, int d2, int d3)`
- **Compute the location of a ball kicked in the given angle and with the given velocity ... after the given time elapsed, starting at (0.0, 0.0)**
  - `// position of the ball kicked with the given velocity after the given time`
  - `CartPt position(double dx, double dy, int t)`
- **Find the position of the given letter in the given String**
- **... produce -1 if not found**
  - `// produce the location of a letter on the given String`
  - `int whereIs(char c, String text)`

Purpose statement, the header of the function/procedure

## Didactics: Algebra and Programing 3

- **Is the sum of three digits divisible by seven?**
  - `divisibleBy7(1, 2, 3) -> false`
  - `divisibleBy7(2, 5, 7) -> true`
- **Compute the location of a ball kicked in the given direction**
  - `position(10.0, 20.0, 3) -> CartPt(30.0, 60.0)`
- **Find the position of the given letter in the given String**
  - `whereIs(“a”, “dedo”) -> -1`
  - `whereIs(“a”, “mama”) -> 1`

Examples of use with the expected outcomes

## Didactics: Algebra and Programing 4

- **Is the sum of three digits divisible by seven?**
  - `inventory: funkcia ‘mod(int) -> int’`
- **Compute the location of a kicked ball ...**
  - `inventory: dx, dy, t; začiatok: (0, 0)`
- **Find the position of a letter in the given String**
  - `inventory: c, text,`

    `funkcia substring(text, i, k)`

-structure -- select fields, record their types
-variants -- process every variant separately
-list all functions/procedures/methods that can be used with the available data

## Didactics: Algebra and Programing 5

- **Is the sum of three digits divisible by seven?**
  [return (d1 + d2 + d3) mod 7 == 0]
- **Compute the location of a ball kicked in the given direction**
  [return new CartPt(t * dx, t * dy)]
- **Find the position of the given letter in the given String**

```
[for (int i = 0; i < length(text); i++)
   if (substring(text, i, i+1) == c)
   return i;
return -1;]
```

Only now work out the body of the function/procedure

## Didactics: Algebra and Programing 6

- **Is the sum of three digits divisible by seven?**
  - `divisibleBy7(1, 2, 3) -> false`
  - `divisibleBy7(2, 5, 7) -> true`
- **Compute the location of a ball kicked in the given direction**
  - `position(10.0, 20.0, 3) -> CartPt(30.0, 60.0)`
- **Find the position of the given letter in the given String**
  - `whereIs(“a”, “dedo”) -> -1`
  - `whereIs(“a”, “mama”) -> 1`

Verify the correctness - the tests are defined in the third step

## Information and data: the connection

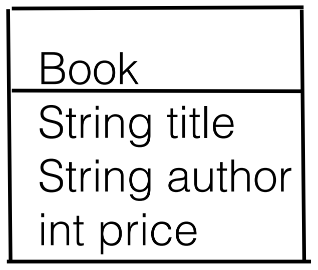
- **primitive/basic types:** numbers, Strings, images, bool
- **clases/structures:** several pieces of data are needed to describe the information
- **references:** a piece of data in one class/structure is an instance of another class/structure
- **variants:** several variants of data share common properties
- **combinations** of these possibilities

Analyze the problem (according to the above criteria), produce data definitions; make several examples of data

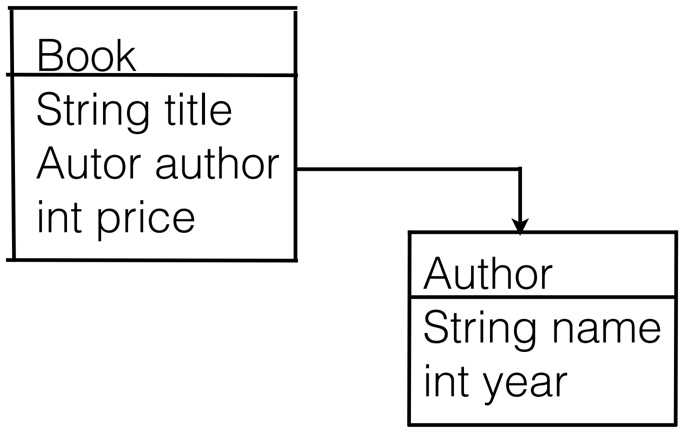
## Information and data: the connection

- **primitive/basic types:** int, String, boolean, image

- **clases/structures:**

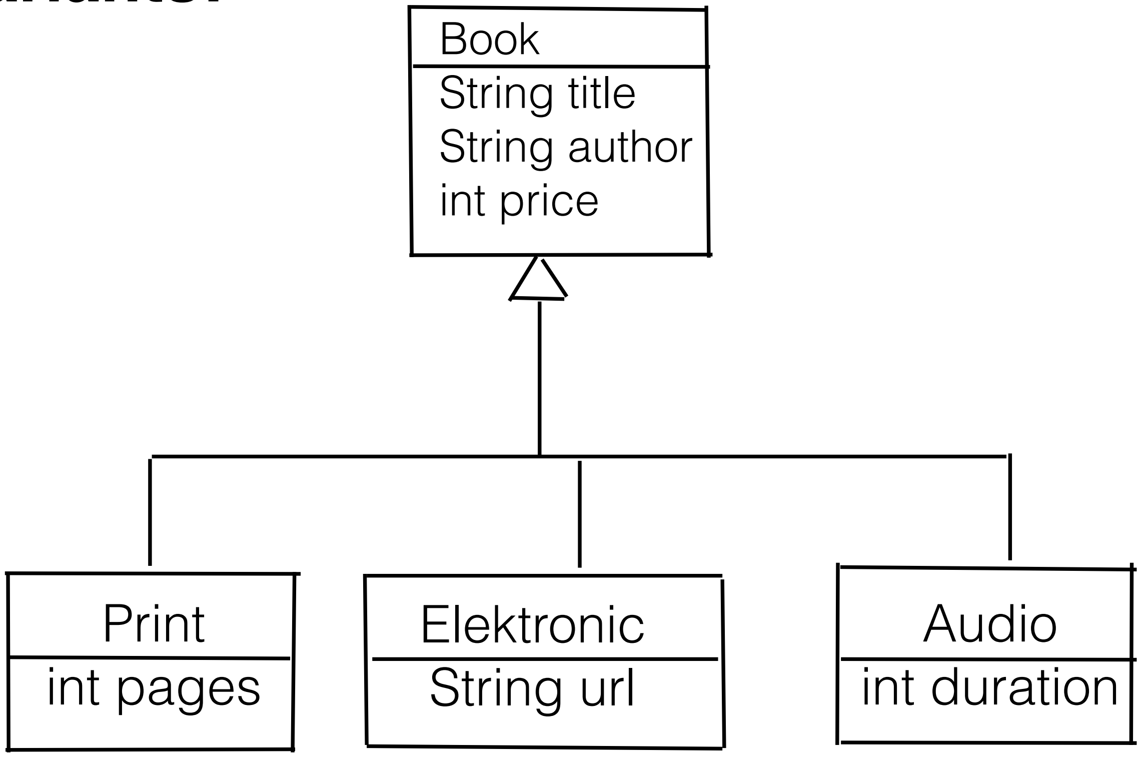


- **references:**



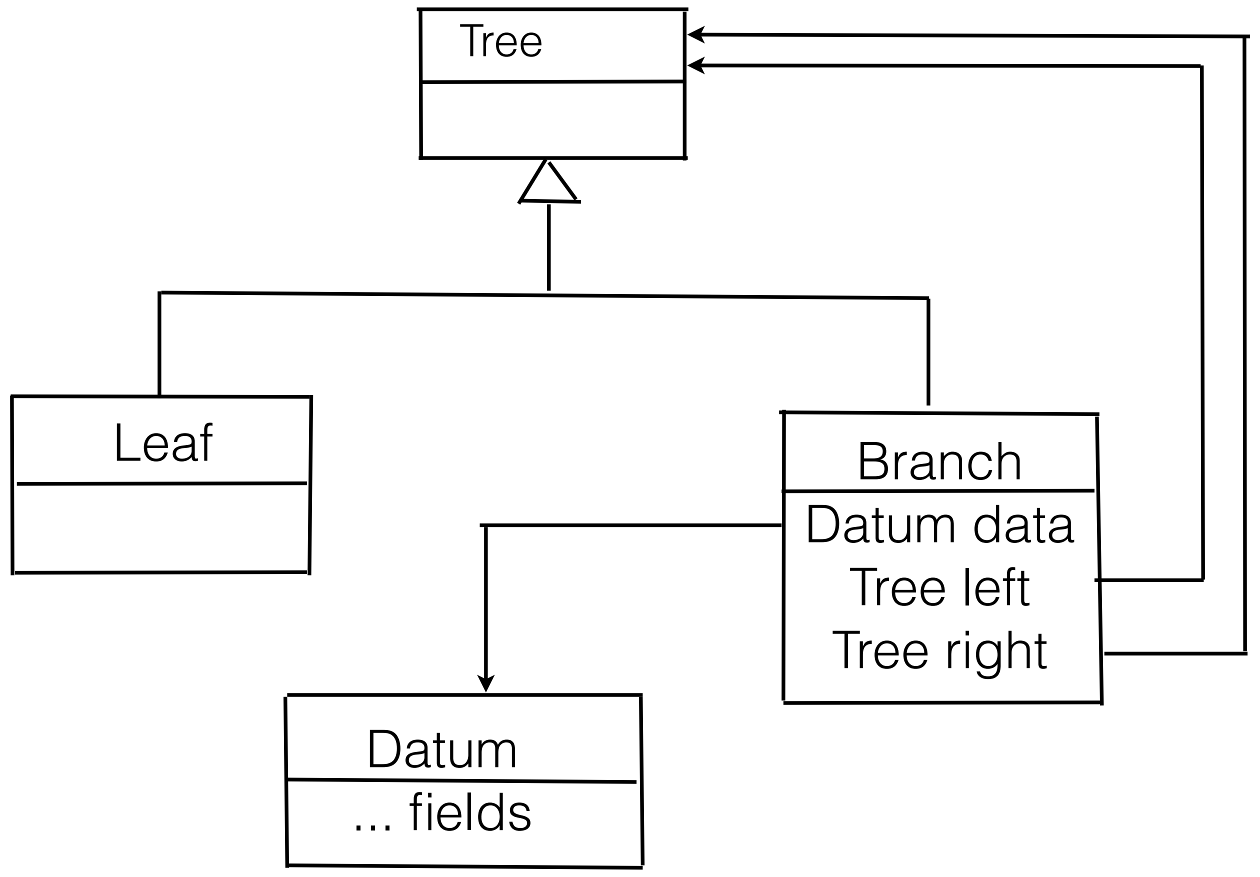
## Information and data: the connection

- **variants:**



## Information and data: the connection

- **combinations:**



## Information and data: the connection

```
                  +-----------+
                  |   Emp     |<-----------------------------+
                  +-----------+                              |
                  |           |                              |
                  +-----------+                              |
                        |                                    |
                        |                                    |
                       /_\                                   |
                        |                                    |
        +-----------------------------------+                |
        |                                   |                |
+----------------+                 +------------------+      |
|    Worker      |                 |    Boss          |      |
+----------------+                 +------------------+      |
|String name     |                 | String name      |      |
|int tasks       |                 | String unit      |      |
+----------------+                 | int tasks        |      |
                          +--------+ ListofEmp peons  |      |
                          |        +------------------+      |
                          |                                  |
                          |                                  |
                          v                                  |
                    +--------------+                         |
                    |  ListofEmp   |<------------------------+--+
                    +--------------+                         |  |
                    +--------------+                         |  |
                           |                                 |  |
                           |                                 |  |
                          /_\                                |  |
                           |                                 |  |
             +--------------------------------+              |  |
             v                                v              |  |
    +-----------------+              +-------------------+   |  |
    |  MTListofEmp    |              |    ConsListofEmp  |   |  |
    +-----------------+              +-------------------+   |  |
    +-----------------+              | Emp first         +---+  |
                                     | ListofEmp rest    +------+
                                     +-------------------+
```

# Information and data: the connection

```
                             +---------+
                             |  Marta  |
                             | CEO 100 |
                             +----+----+
                                  |
                +-----------------+-----------------------------+
                |                                               |
        +-------+-----+                                     +---+---------+
        | Danko       |                                     | Anka        |
        | Operacie 70 |                                     | Financie 20 |
        +------+------+                                     +------+------+
               |                                                   |
   +-----------+-------------------+------------+               +--+------+
   |                               |            |               |         |
+-------------+            +-------+-----+ +----+--------+   +-----+ +----+--------+
| Jurko       |            | Janka       | | Palko       |   | A 3 | | Peter       |
| Sekcia-A 25 |            | Sekcia-B 15 | | Sekcia-C 20 |   +-----+ | Sekcia-D 10 |
+----+--------+            +-----------+-+ +----------+---+           +----+--------+
     |                                 |              |                    |
  +--+-------------+-------+         +-+-----+      +-+---+             +--+---+
  |                |       |         |       |      | H 6 |             | B  5 |
+-+------------+ +-+---+ +-+---+   +-+---+ +-+---+  +-----+             +------+
| Milan        | | D 4 | | E 2 |   | F 8 | | G 6 |
| Skupina-A 10 | +-----+ +-----+   +-----+ +-----+
+--+-----------+
   |
+--+--+
| C 6 |
+-----+
```

# Information and data: the connection

```
                             +---------+
                             |  Marta  |
                             | CEO 100 |
                             +----+----+
                                  |
                +-----------------+-----------------------------+
                |                                               |
        +-------+-----+                                     +---+---------+
        | Danko       |                                     | Anka        |
        | Operacie 70 |                                     | Financie 20 |
        +------+------+                                     +------+------+
               |                                                   |
   +-----------+-------------------+------------+               +--+------+
   |                               |            |               |         |
+-------------+            +-------+-----+ +----+--------+   +-----+ +----+--------+
| Jurko       |            | Janka       | | Palko       |   | A 3 | | Peter       |
| Sekcia-A 25 |            | Sekcia-B 15 | | Sekcia-C 20 |   +-----+ | Sekcia-D 10 |
+----+--------+            +-----------+-+ +----------+---+           +----+--------+
     |                                 |              |                    |
  +--+-------------+-------+         +-+-----+      +-+---+             +--+---+
  |                |       |         |       |      | H 6 |             | B  5 |
+-+------------+ +-+---+ +-+---+   +-+---+ +-+---+  +-----+             +------+
| Milan        | | D 4 | | E 2 |   | F 8 | | G 6 |
| Skupina-A 10 | +-----+ +-----+   +-----+ +-----+
+--+-----------+
   |
+--+--+
| C 6 |
+-----+
```

How many people work in the unit with the given name?

## What is important:

- There are many types of data, we can combine them if the information consists of several types or variants
- The goal pf a program is to produce from the given data some new data that represents new information
- Every fnction or procedure should handle just one task: use helper functions/methods when needed
- Input-output is not important on its own: processing of the inputs and preparation of data for output are tasks to be programmed as well

## What is important:

- It is important to know how to represent information as data and how to interpret data as the information it represents
- Every function/procedure should be designed systematically, test-first approach...
- Build larger programs by designing abstractions

    if code is repeated (with only small differences), produce a program where the differences are represented by parameters and the common part appears only once

## Design of abstractions = libraries

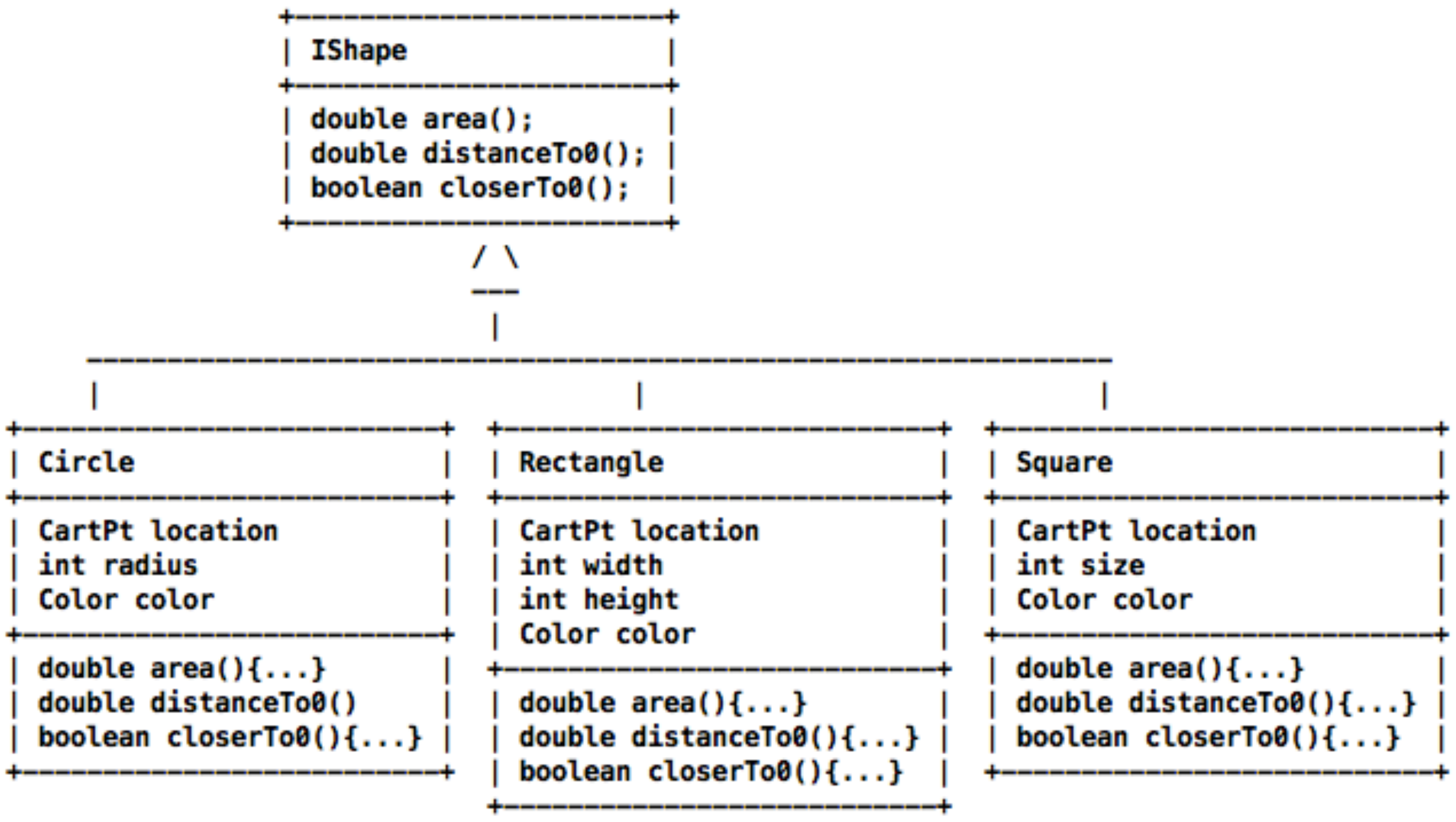
- Mark all places where the similar code segments differ.
- Replace them with parameters and rewrite the solution using them as arguments.
- Rewrite the original solutions to your problems by invoking the generalized solution with the appropriate arguments.
- Make sure that the tests for the original solution still pass.

## Design of abstractions = libraries

- interfaces -- abstract classes
- function objects
- parametrized types
- traversals
- abstract data types

The keys to understanding how to build/use reusable code

## Design of abstractions: abstract class



```
                +-----------------------+
                | IShape                |
                +-----------------------+
                | double area();        |
                | double distanceTo0(); |
                | boolean closerTo0();  |
                +-----------------------+
                           / \
                           ---
                            |
     -------------------------------------------------------------
     |                                 |                              |
+---------------------------+ +-----------------------------+ +-----------------------------+
| Circle                    | | Rectangle                   | | Square                      |
+---------------------------+ +-----------------------------+ +-----------------------------+
| CartPt location           | | CartPt location             | | CartPt location             |
| int radius                | | int width                   | | int size                    |
| Color color               | | int height                  | | Color color                 |
+---------------------------+ | Color color                 | +-----------------------------+
| double area(){...}        | +-----------------------------+ | double area(){...}          |
| double distanceTo0()      | | double area(){...}          | | double distanceTo0(){...}   |
| boolean closerTo0(){...}  | | double distanceTo0(){...}   | | boolean closerTo0(){...}    |
+---------------------------+ | boolean closerTo0(){...}    | +-----------------------------+
                              +-----------------------------+
```

Create an abstract class: common fields, methods

## Design of abstractions: abstract class

```
                 +----------------------------+
                 | AShape                     |
                 +----------------------------+
                 | CartPt location            |
                 | Color color                |
                 +----------------------------+
                 | abstract double area();    |
                 | double distanceTo0(){...}  |
                 | boolean closerTo0(){...}   |
                 +----------------------------+
                              / \
                              ---
                               |
     ---------------------------------------------------
      |                          |                       |
+-----------------------+  +---------------------+  +---------------------+
| Circle                |  | Rectangle           |  | Square              |
+-----------------------+  +---------------------+  +---------------------+
| int radius            |  | int width           |  | int size            |
+-----------------------+  | int height          |  +---------------------+
| double area(){...}    |  +---------------------+  | double area(){...}  |
| double distanceTo0()  |  | double area(){...}  |  +---------------------+
+-----------------------+  +---------------------+
```

Create an abstract class: common fields, methods

## Design of abstractions: function objects

- sort a collection of the type T (what **ordering**?)

  comparator function: (T, T) -> int

- select all items that fit some **criterion**

  selector predicate: (T) -> boolean

- perform the desired **action**

  action method to perform

*Difficult if the language does not support first class functions*

## Design of abstractions: parametrized types

- binary search trees of numbers, strings, books

  Tree<T> --> Tree<Integer>, Tree<String>, ...

- lists of persons, songs, images

  List<T> --> List<Person>, List<Song>, ...

*Not needed in untyped languages*

## Design of abstractions: traversals

Some algorithms require that we examine all elements of a data set, one at a time.

Iterators, visitors, and specially designed classes or language constructs provide such service.

The algorithms then rely on these services to generate the needed data items, without the knowledge of how the data set is implemented.

Iterator may generate data from an array or from a file...

## Design of abstractions: abstract data types

- Vector (ArrayList) -- direct access structures
- Stack, Queue
- Priority Queue
- Map, HashMap -- (key - value) pairs
- Graph

See several implementations -- understand the trade-offs

## Libraries for the beginners: Java

- Typical programming languages are not suitable for a beginner
- They contain features that the beginner does not understand, but the error messages refer to them
- Programming of inputs, outputs, and user interactions is difficult and beginner needs to learn a lot before he is ready to program them
- Design of tests requires understanding of the different ways of evaluating equality of data

## Libraries for the beginners: Java

Programing language for the beginners: **FunJava**

- Every class can implement only one interface
- Every field must get initial value when defined in the class or when the constructor is invoked
- The value of the field never changes
- The language has only two statements:

  return expression

  if (condition) statement else statement

Typical programing language is not suitable for a beginner

## Libraries for the beginners: Java

Programing language for the beginners: **FunJava**

- Every class can implement only one interface
- Every field must get initial value when defined in the class or when the constructor is invoked
- The value of the field never changes
- The language has only two statements:

  return expression

  if (condition) statement else statement

Contains features that the beginner does not understand, but the error messages refer to them - not a problem here

## Libraries for the beginners: Java

Library for the beginners: **World, Canvas**

- simple functions for drawing of geometric shapes (circle, disk, rectangle, line, text)
- **World:** (programming of interactive games)

    Canvas theCanvas -- accessible field
        where the game scene is drawn

    boolean draw() -- method that draws the
        scene

Programming of inputs, outputs, and user interactions is difficult and beginner needs to learn a lot before he is ready to program them

## Libraries for the beginners: Java

Library for the beginners: **World, Canvas**

-**World:** (programming of interactive games)

- actions:

  World onTick()

  World onKeyEvent(String ke)

  boolean endOfWorld()

- world begins the animation by invoking

  bigBang(int width, int height, double tick)

Programming of inputs, outputs, and user interactions is difficult and beginner needs to learn a lot before he is ready to program them

## Libraries for the beginners: Java

Library for the beginners: **World, Canvas**

- **World:** (programming of interactive games)

  for advanced programmers

  mouse actions

  sound (MIDI notes to play on tick/key

  (programing of sequences of notes and their combinations)

  universe: client-server with messages

Programming of inputs, outputs, and user interactions is difficult and beginner needs to learn a lot before he is ready to program them

## Libraries for the beginners: Java

Library for the beginners: **Tester**

- special library for the design and evaluation of tests
- compares two objects by their value, not by reference
- compares inexact values within the given tolerance
- special tests for constructors, exceptions, iterators, one-of options, value within a range

Programming of inputs, outputs, and user interactions is difficult and beginner needs to learn a lot before he is ready to program them

## Libraries for the beginners: Java

Library for the beginners: **Tester**

- produces a report with all results:
  - prints the values of all objects (pretty-print)
  - produces the results of all tests
  - if the test fails,
    - shows side-by-side the actual and expected values
    - marks the first place where the values differ
    - provides a link to the failed test

Programming of inputs, outputs, and user interactions is difficult and beginner needs to learn a lot before he is ready to program them

## What is important:

- Instead of just using libraries teach students how libraries are built

  function(objects) that compare data

  algorithsm that use function(objects) (sort, filter, andmap)

- Abstract data types and their implementation
- Foundations of evaluation of complexity of algorithms and data structures

## What is important:

- Principles of processing data from inputs and generating data for outputs:
  - conversion of a String to numeric value it represents
  - encoding of data when saved in files
  - event handling - principles, and their use
  - input and output streams
- Principles of design:
  - test-first design
  - one task - one function/procedure

## Conclusion

### Thank you for listening

**Program by Design:**

**http://www.programbydesign.org**

**Java libraries:**

**http://www.ccs.neu.edu/javalib**

**Laboratories, materials:**

**http:://www.ccs.neu.edu/home/vkp/Teaching**

**Curriculum for 6-9 grade:**

**http://www.bootstrapworld.org**