Practice of Programming using Java

Lecture 1
June 15, 2006
6-8pm LT

Objectives

- Improve/Learn Programming
  - Basics of programming
  - Organizing and using data structures and control structures
- Introduce Object Oriented Programming
- Learn Java as a programming language
- Introduction to good programming practices

Schedule

- Saturdays: 2 to 5 pm
- Tuesdays: 6 to 8 pm
- Thursdays: 6 to 8 pm
- Fridays: 6 to 8 pm
- Last lecture: July 5th.
- Exam: July 9th 9 to 12 AM
- Any changes will be emailed to you.
- See course webpage: http://www.cse.iitb.ac.in/~rkj/summercourse
- TAs will be available as per allocation.

Note your TAs

- raghuvar@cse — mostly in H6/276 or OSL
- aniketd@cse — H6/12 but mostly in OSL(maths1)
- padmaja@cse — mostly in CH Kresit
- dhananjays@cse — mostly in H13 B/222 or OSL
- chawley@iitb — H5/113, but mostly in met 3rd floor (302 Met) polymer lab
- saurabhg@ee — H4/106 mostly in GG bldg, 5th floor- design lab
- manishg@cse — H12/C508. but mostly in OSL(maths1)
- Use email to contact them, and use 7730 or email to contact me.
- Assignment submissions will be through the website.
- The course schedule is fast paced. Regular practice and programming will be required to get the best (grade:-) out of the course.

Today's Topics: Basic Ideas

- Program
- Data
- Control
- Interfaces
- Objects
- Classes

A Program

- Pickup 100 Rs.
- Go to market
- Select a vegetable vendor
  - Choose a vegetable, assess its quality, ask for its price, if satisfied select a quantity and buy it.
  - Repeat till you are satisfied/have money
- Repeat the above with another vendor to complete your vegetable collection.

A Program

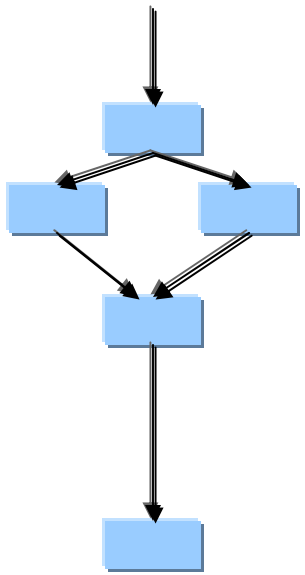
- Control Logic
- Data

At runtime:

- Initialization
- Dynamic changes to data as per the control logic
- Termination condition

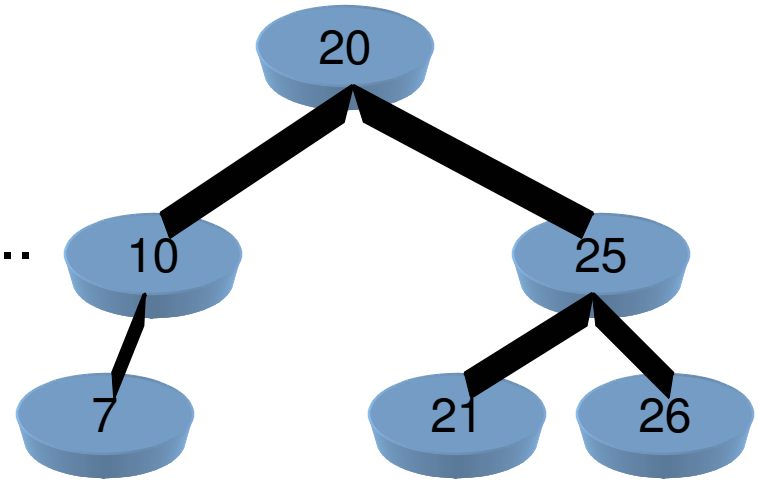
A quick look at some Control Abstractions (we shall visit this again)

- Basic Control Abstractions
  - Functions, function calls, recursion
  - Assignment statement
  - Sequential execution
  - If then else, case statements
  - While, repeat, case and for statements
  - Threads
- Control abstractions cover data flows and value changes → i.e. dynamics



A quick look at some Data Abstractions (we shall visit them again)

- Data Abstractions
  - symbols and lists
  - Types: int, bool, char, float..
  - Structures
  - Unions, enumerated types
  - Arrays, Vectors

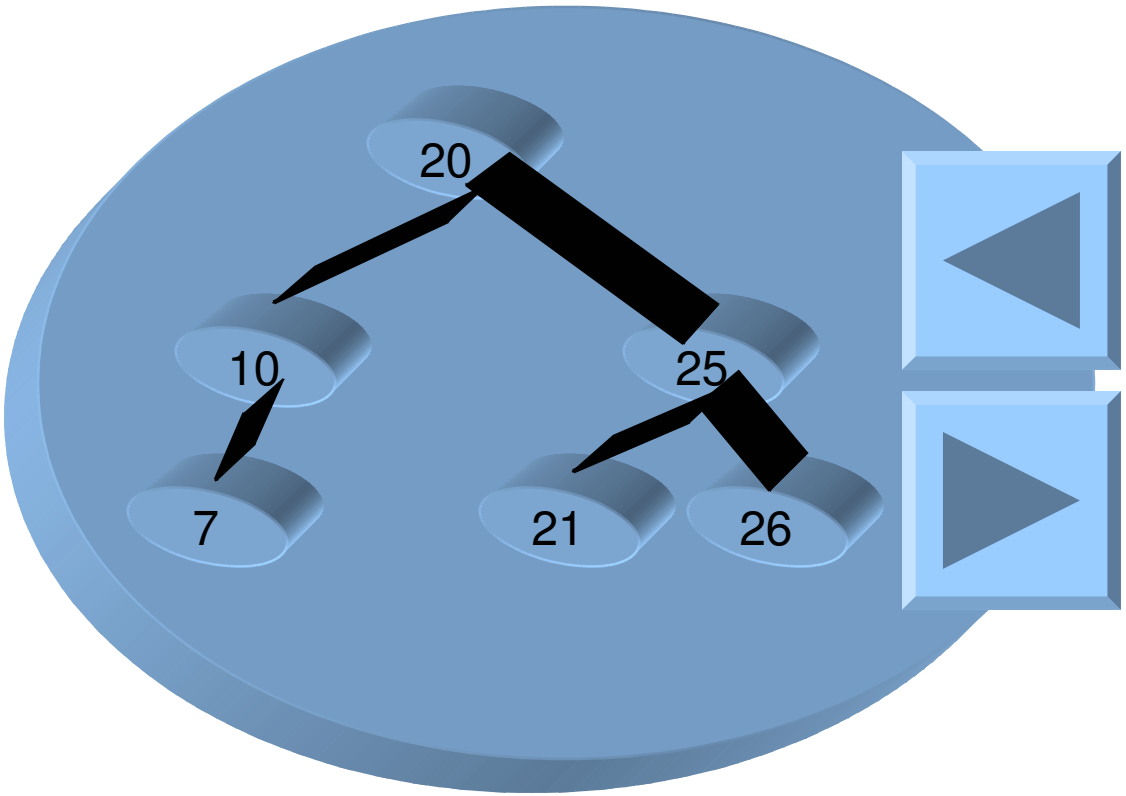


- Data abstractions are entity abstractions
- operations supported on data abstractions are mostly general: read, write

Richer Abstractions

- The above control and data abstraction are low level abstraction as compared to…
- High level abstractions need to be composites of these
  - Besides function composition, structures:
  - it makes sense to combine data and control together to form an interesting composite abstraction called **object**

Object Abstraction



Add new Element

Remove an Element

Examples of Richer Abstractions used inside your PC.

- File at OS level
  - Data: stream of bytes
  - Operations supported: open, close, read, write, rewind, seek
- Process at OS level
  - Data/internal objects: control and data segments, page tables, open files, priority..
  - Control: create, terminate, suspend, resume, trace
- Stack Data structure
  - Data: elements arranged in the form of stack
  - Control: create, delete, push, pop, top

More Examples of Richer Abstractions which you may have used directly or indirectly

- Table in a spreadsheet/GUI
  - Data: rows, columns, content
  - Operations (control): create, delete, add/del row/column, insert element
- Name server
  - Data: name-location bindings arranged in a hierarchy
  - Operations: add new binding, delete existing binding, create/delete namespaces

Compare These with Some Examples of Abstraction in Real life

- Fan
  - Data/internal objects: motor, capacitor ..
  - Operations: switch on, off, set speed
- Tape
  - Data/internal objects: internal circuits, cassette holder
  - Operations: switch on/of, open/close cassette holder, play, rewind, forward, record, pause, continue
  - It's a composite object: player/recorder + cassette holder

They have something in common:
→ Explaining Object Abstractions

- It is convenient to think of abstractions in terms of the data that they possess along with the operations which they allow on them
  - Data: Internal data elements, internal objects
  - Operations: Expose for External Use
- User only worries about how to use an abstraction but now how it is implemented

The important steps in formulating object abstractions

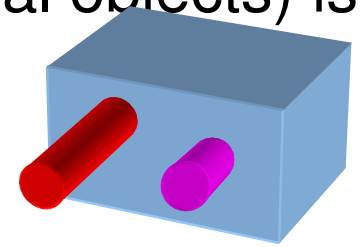
- Thinking data and high level control on the data as one unit of abstraction
- Separating internal data from exposable operations on them (i.e. separation of Interfaces from Implementation)
- Hiding data from external environment (through Encapsulation)

Interfaces and Implementation

- Interactions with an object from an external entity happens through the object's interface (s). An interface is a collection of object's controls.
- Implementation is responsible to provide the required behavior
- An implementation can be changed so long as it sticks to what is promised as part of interface

Two Basic Principles of Object Orientation

- Abstraction
  - Hiding complexities of lower layer
  - We discussed about data and control abstractions
  - Object abstraction: data + observable behavior together as a unit
- Encapsulation
  - Only the observable behavior is exposed, the rest (data and internal objects) is hidden from external environment



Exercise Session 1

- Define one of the following objects in terms of their observable behavior as Java interfaces.
  - Stack of books
  - List of items
  - Push button

  (expected code size: at most half a page)

Exercise Session 2

- Figure out the object abstractions of the below objects. Express their interfaces and internal components in Java.
  - Washing Machine
  - Bike
  - MP3 CD Player
  - Phone unit on your desk

  (at least one example per student. Pick from above list, or pick some other real life object abstraction.).

  (expected code size: 1 page)

A Class and its instance

- Class
  - Defines the structure
    - Data
    - Behavioral abstraction
- Instance
  - The actual object that is to be used
  - Has values which get updated
  - You can invoke member functions on it
  - Many instances of same class can be created

Exercise Session 3

- Write class specifications for one of the below objects, implement the class fully and test it through a main() function.
  - Tossable Coin
    - Achieve the randomness
  - Coffee Vending Machine
    - Implement a few rules

  Alternatively, you can implement any other interesting (but simple) object if you wish to.

  (expected code size: 1 page)