Object Oriented Programming in Java
Week 1

OOP Concepts

- Object, Class, Type
- Methods, Messages
- Behavior-based Design
- Class library, hierarchy
- Claims for Java
- Reading Budd, Ch 1-3, 4

Abilities & Skills

- Read a design
- Design a small program
- Extract a design
- Run a VAJ program
- Change that program, using the class library
- import/export VAJ code

- **Asst** (due next Mon)

  represent the design for ClickMe as CRC Cards (lab pair hand in)

  do programs and exercises for Ch. 4 (individual, no hand in)

Object, Class, Type

- Object - an encapsulation of state and behavior
- Attributes
- Class, Instance, Hierarchy
- Abstract classes
- Information hiding

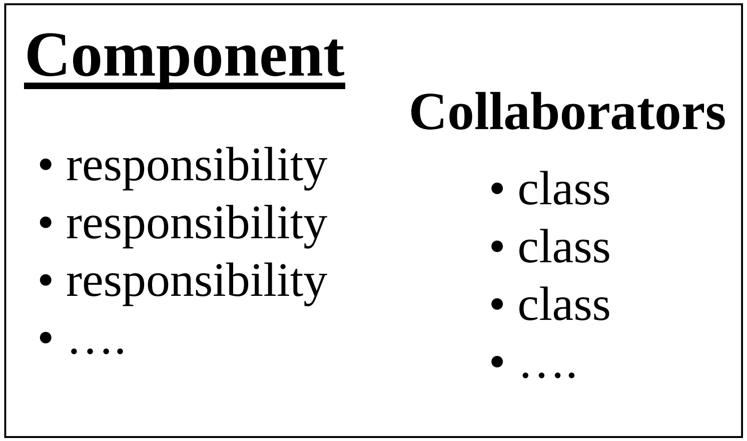
Methods, Messages Behavior

- How different from a procedure call
- Method binding
- Polymorphism
- Method overriding
- Exception handling

Responsibility-Driven Design

“The major problem in software development:
management of details and communication of information
between diverse portions of the project.”

- Delegate to a class, responsibility for a task
- Components & Reuse
- Component-Responsibility-Collaborators



Component

Collaborators

- responsibility
- responsibility
- responsibility
- ….

- class
- class
- class
- ….

Interactive Kitchen Helper

- A PC based application that replaces index-card recipes and assists in meal planning for a period of time (e.g., a week).
  - User actions
    - Browse recipe database
    - Create menus
  - System services
    - Scale recipes for number of servings
    - Print out menus for a week, day or meal
    - Print grocery list

Work with 5 others to finish the IKH CRC design

(each person play role of one component)

(see p. 39)

CRC Cards for Interactive Kitchen Helper

Greeter

- Display initial msg
- Offer option choice
- Pass control
  - DB Mgr
  - Plan Mgr

Collaborators

- DB Manager
- Plan Manager

Date

Maintain info about specific date

- date (year,month, day) -- create new date
- displayAndEdit() -- display date info in window, allow user to edit entries
- buildGroceryList (List&) -- add items from all menus to grocery list

Collaborators

- PlanManager
- Meal

Plan Manager

•

Collaborators

- Date

Meal

•

Collaborators

Recipe Database

•

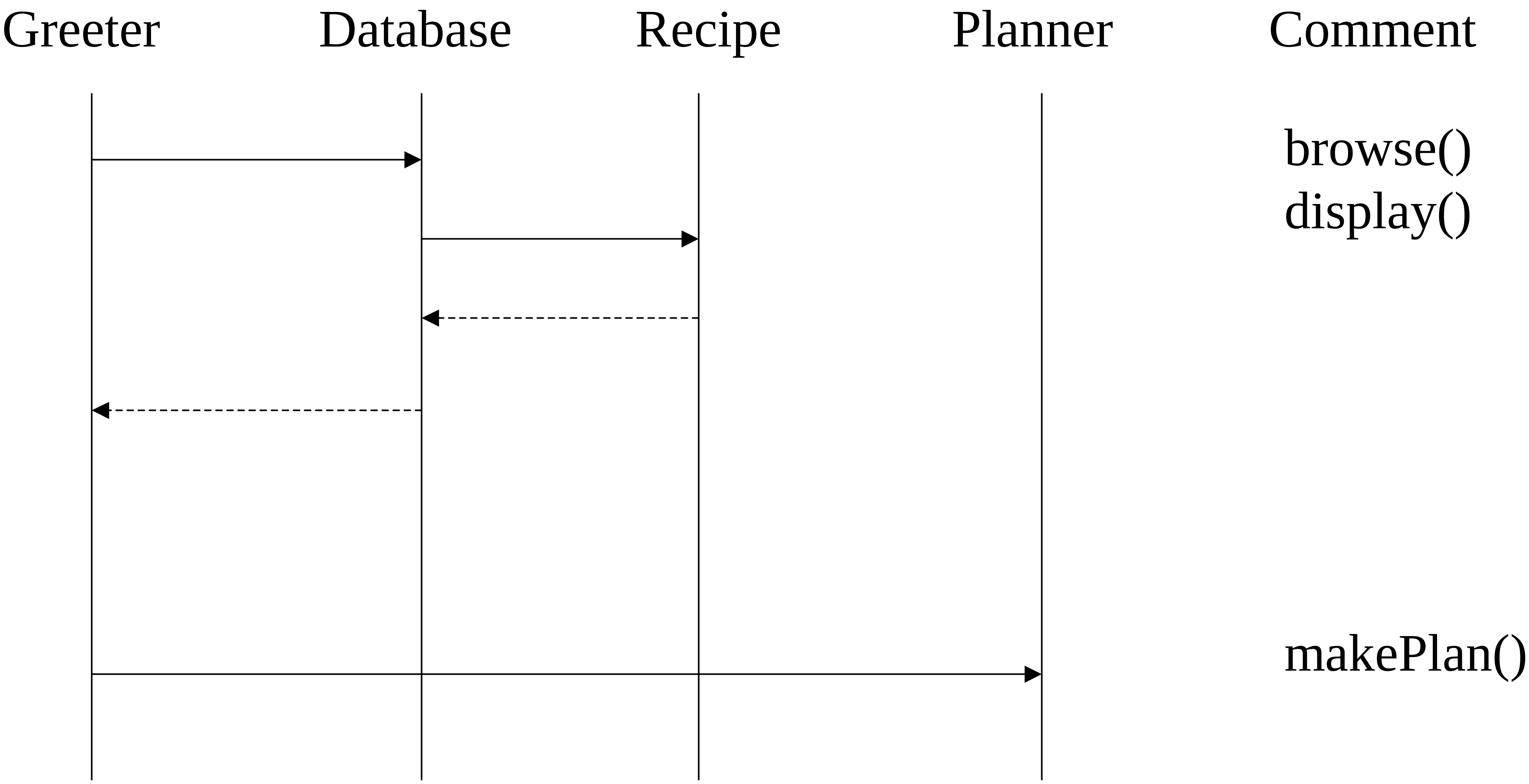
Collaborators

Recipe

•

Collaborators

Interaction Diagrams



Greeter Database Recipe Planner Comment

browse()

display()

makePlan()

Design Process

- Characterize behavior of the entire application.
- Refine the specification.
  - “look and feel”
  - Determine high level structure, mapping activities onto components
- Identify high level system components
- What-who cycle.
- Scenarios.
- Documentation & user manual
- system design documentation
- arguments for/against design alternatives
- project schedule and project diary (status reports)
- formalize the interface
- Define information required by and maintained within each component
- Review components and interface names.
- Assign 1+ component to team members to design data structure(s) and algorithm(s) for each method.
- Implementation
- Identify “facilitator” components.
- Characterize & document component preconditions;
- Unit test.
- Integration.

Design and Development Hints

- When is a component “too big” or small?
- Anticipate change
- Isolate s/w from hardware
- Reduce coupling
- Component should have high cohesion
- Maintain records of design process
- Names should be [Keller 1990]:
- Integrate iteratively, using stubs.
- Use regression testing.

Is Java the “silver bullet”?

- Client-side computing
- Simple
- O-O to the core
- Network savvy.
- Interpreted.
- Robust.
- Secure.
- Architecture neutral.
- Portable.
- High performance
- Multithreaded
- Dynamic

Be prepared to argue for or against these
with specific language features and code

In Lab Tomorrow….10-12

- Work with your partner in the ACC
- Gain familiarity with VAJ, take home a copy
- ClickMe (w/ modifications)
  - advanced students to start on BallWorld in VAJ
- Assignment due Monday -- CRC cards for ClickMe

Java Program Structure

public static void main (String [ ] args) { }

access modifiers *method name* *type*

return type *arguments*

```
public class BankAccount {
      private int accountBalance = 0;
public void display() {
      System.out.println
("the current account balance is " + accountBalance);
      }
public void main(String[] args) {
      deposit (15);
      withdrawal (10);
      display();
      // System.out.println(accountBalance);
      }
      public void withdrawal (int amount) {
            accountBalance = accountBalance - amount;
```

A simple class -- BankAccount

```
public class BankAccount {
  private int accountBalance = 0;
public void display() {
  System.out.println
("the current account balance is " + accountBalance);
  }
public void main(String[] args) {
  deposit (15);  withdrawal (10);  display();
  }
public void withdrawal (int amount) {
      accountBalance = accountBalance - amount;
  }
}
```