Chapter 2
Data abstraction:
introductory concepts
Values and types

- **value**: fundamental pieces of information that can be manipulated in a program.
- **type**: a set of related values along with the operations that can be performed with them.
Values

- 2 kinds of values:
  - *simple* (*atomic, primitive*): integers, characters
  - *composite*: date(day, month, year), string(several characters)

- Values are abstractions used to model various aspects of a problem.
Types

- Values grouped together according to how we use them and what operations we perform on them.
- Example: Integers use $+,-, <, >$, etc.
Primitives

- Built-in types.
- **Java primitives**: byte, short, int, long, float, double, char, and boolean.
- Java has no built-in composite primitives.
Numbers

- **Integers:**
  - `byte` (1 byte)
  - `short` (2 bytes)
  - `int` (4 bytes)
  - `long` (8 bytes)

Used for counting (whole numbers).

<table>
<thead>
<tr>
<th>Type</th>
<th>Smallest value</th>
<th>Largest value</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>-128</td>
<td>127</td>
</tr>
<tr>
<td>short</td>
<td>-32,768</td>
<td>32,767</td>
</tr>
<tr>
<td>int</td>
<td>-2,147,483,648</td>
<td>2,147,483,647</td>
</tr>
</tbody>
</table>
Numbers (cont.)

- **Floating point:**
  - float (4 bytes)
  - double (8 bytes)

- Used to represent things we measure (anything with a decimal point).

- The computer’s representation may not be exact.
Characters

- `char` includes upper case and lower case letters, digits, punctuation marks, and other special characters.
- Characters contained in the *Unicode* character set.
- Used for input/output.
Boolean

- Only 2 values: true or false.
Objects

- The fundamental abstractions from which we build software systems.
- Designed to support the *functionality* of the system.
- Responsible for performing specific tasks.
Queries and properties

- Furnish relevant information about what the object represents.
- **properties**: characteristics about which the object provides information.
- Each property has an associated **value**.
- **queries**: requests for information concerning properties.
Commands and state

- The values of properties can change at any time.
- **state**: the set of an object’s properties and associated values at any given time.
Commands and state (cont.)

- Commands instruct the object to perform some action which often results in the object changing state.
- At any given time, a particular value is associated with each property of an object.
EXAMPLE SYSTEM USING OBJECTS

University registration System

What are the objects?

Student  Course
Queries:

Student
- What is your name?
- How many credit hours?
- What are you enrolled in?
- Have you paid your fees?

Course
- What is the course number?
- How many credit hours is it?
- What room is the course held in?

Properties:

name
address
ssn
credit hours
major
year in school
course number
section
professor
room
time/day
total seats
seats taken
States:

Student

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>R. Raskolnikov</td>
</tr>
<tr>
<td>address</td>
<td>S. Place, Petersburg</td>
</tr>
<tr>
<td>social security number</td>
<td>000-00-0001</td>
</tr>
<tr>
<td>credit hours</td>
<td>9</td>
</tr>
<tr>
<td>fees paid</td>
<td>no</td>
</tr>
</tbody>
</table>
| course schedule | Ethics 1001
                             Law 6592
                             Comp Sci 1583

Course

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>CS 101</td>
</tr>
<tr>
<td>room</td>
<td>DSB 146</td>
</tr>
<tr>
<td>professor</td>
<td>Dr. Smith</td>
</tr>
<tr>
<td>credit hours</td>
<td>3</td>
</tr>
<tr>
<td>prerequisite</td>
<td>CS 98</td>
</tr>
</tbody>
</table>

Commands:

• add a course
• drop a course
• change the address
• determine fees
• change the room number
• set professor
• calculate the number of seats available
States and commands

**Student**

- credit hours = 9
- fees paid = no
- course schedule = Ethics 1001, Law 6592, Comp Sci 1583

**Student**

- credit hours = 6
- fees paid = no
- course schedule = Law 6592, Comp Sci 1583

State before the command

command

State after the command

Drop course (Ethics 1001)
Designing with objects

- What are the objects?
- What features should these objects have?
Particular object responsibility

- What must an object know?
  - The properties of the entity the object is modeling, and
  - About other objects with which it needs to cooperate.
Particular object responsibility (cont.)

What must an object do?

- Compute particular values.
- Perform actions that modify state.
- Create and initialize other objects.
- Control and coordinate the activities of other objects.
Particular object responsibility (cont.)

- Asking what an object knows (what makes it what it is): *object queries*.
- Asking what an object does: *object commands* and *queries*.
Knows about self:

**Student**
- Student’s name
- Student’s address
- Student’s social security number
- Number of credit hours enrolled

**Course**
- Course’s title
- Course’s section
- Course’s professor
- Course’s time
- Course’s room number

**Does (Commands):**

**Student**
- Add a course to the student’s schedule
- Drop a course from the student’s schedule
- Calculate student’s fees

**Course**
- Change room assignment
- Add a student to the roll
- Print a class roll
- Decide if the class is full
Does(Queries):

**Student**
- Supplies a student’s name. Query: What is the student’s name?
- Supplies a student’s current classes.
- Supplies the student’s address

**Course**
- Supplies the Professor of the class. Query: Who is teaching this class?
- Supplies the number of student’s enrolled in the class.
Ticket dispenser example

- Queries:
  - How many tickets have we sold?
  - How many tickets are left?

- Properties:
  - Next ticket number
  - Tickets left
Ticket dispenser example (cont.)

States and Commands

Ticket Dispenser

next ticket = 27
tickets left = 43

$\text{time}_0$

state before the command

Ticket Dispenser

next ticket = 28
tickets left = 42

$\text{time}_1$

state after the command

Dispense ticket

command
Chess example

- Queries:
  - Where is a certain piece?
  - What color is a player’s pieces?
  - Is a player in check?

- Properties:
  - Player’s color
  - Piece’s position
Chess example (cont.)

States

- User
  - user:Player
    - color = white
    - play = from user
  - Board...
  - white queen:Piece
    - color = white
    - name = q
    - inPlay = yes
    - position = c5
  - computer:Player
    - color = black
    - play = automatic
Chess example (cont.)

- Commands:
  - Move a piece?
  - Change whose turn it is?
Classes

- An object is an instance of a class.
- A class specifies the features of a group of similar objects.
- Two objects from the same class will have the same set of queries and commands, but usually contain different values.
Objects as properties of objects

- A relation associates two or more objects in a specific, well defined way.
- Relations define how objects interact to solve a problem.
- An object’s property can be a reference to another object.
Reference Examples

Person
- name = Elizabeth Bennet
- age = 22
- sex = female
- mother =

Person
- name = Mrs. Bennet
- age = 48
- sex = female
- mother = ...

Board
- a1 =
- a2 = ...
- ...

white queen: Piece
- color = white
- name = q
- inPlay = yes
- position = a1
Reference

- A value that denotes an object
- *null reference*: refers to no object.

```
Person

name = Adam
age = 4086
sex = male
mother = 
```
We’ve covered

- Introduction of fundamental notions of values and objects.
- Java provides:
  - Integer types (`byte`, `short`, `int`, `long`)
  - Real or floating type (`float`, `double`)
  - Character type (`char`)
  - Boolean type (`boolean--true/false`)
Glossary

class: a collection of similar objects. Objects that are members of the same class have the same set of features.

command: a constituent (feature) of an object used to instruct the object to perform some action which typically results in a change of the object’s state.

doing responsibility: a requirement that an object be able to accomplish some task.

feature: a query or command to which an object responds.

immutable object: an object that does not change state after it is created.

instance: an object. An object that is a member of a class is called an “instance of the class.”

knowing responsibility: a requirement that an object know some piece of information.

null reference: a unique reference value that doesn’t denote any object.

primitive type: a type provided as an integral part of the language. Java’s built-in types include byte, short, int, long, float, double, char and boolean.

property: a characteristic of an object. A value is associated with each property of an object.
query: a constituent (feature) of an object used to determine the current value of one of the object’s properties. There is one query feature for each property of an object.
Also means use of the feature: “to query an object” means to obtain the current value of a property by using a query feature.

reference value: a value which denotes or refers to an object.

relation: a specific, well-defined association between two or more objects.

responsibility: something required of an object. Generally, a requirement that an object know some information (“knowing” responsibility) or be able to accomplish some task (“doing” responsibility).

state: the set properties of an object and their current values.

system functionality: the set of tasks a system is required to be able to perform.

type: a set of similar values along with the operations that can be performed with them.

value: a fundamental piece of information manipulated in a program.