Exercises 3.1 - 3.4
Object Containment and Methods

Exercise 3.1

Develop a "real estate assistant" program. The "assistant" helps the real estate agent locate houses of interest for clients. The information about a house includes its kind, the number of rooms, the asking price, and its address. An address consists of a house number, a street name, and a city.

Represent the following examples using your classes:

- Ranch, 7 rooms, $375,000, 23 Maple Street, Brookline
- Colonial, 9 rooms, $450,000, 5 Joye Road, Newton
- Cape, 6 rooms, $235,000, 83 Winslow Road, Waltham

Exercise 3.1 (cont)

Develop the following methods for the class House:

1. `hasMoreRooms`, which determines whether one house has more rooms than some other house;
2. `inThisCity`, which checks whether the advertised house is in some given city (assume we give the method a city name);
3. `sameCity`, which determines whether one house is in the same city as some other house.

Solution

Exercise 3.2

. . . Develop a program that assists bookstore employees. For each book, the program should track the book's title, its price, its year of publication, and the author. A author has a name and birth year.

- Develop the following methods for this class:
  - `currentBook` that checks whether the book was published in 2004 or 2003;
  - `currentAuthor` that determines whether a book was written by a current author (born after 1940);
  - `thisAuthor` that determines whether a book was written by the specified author;
  - `sameAuthor` that determines whether one book was written by the same author as some other book;
  - `sameGeneration` that determines whether two books were written by two authors born less than 10 year apart.

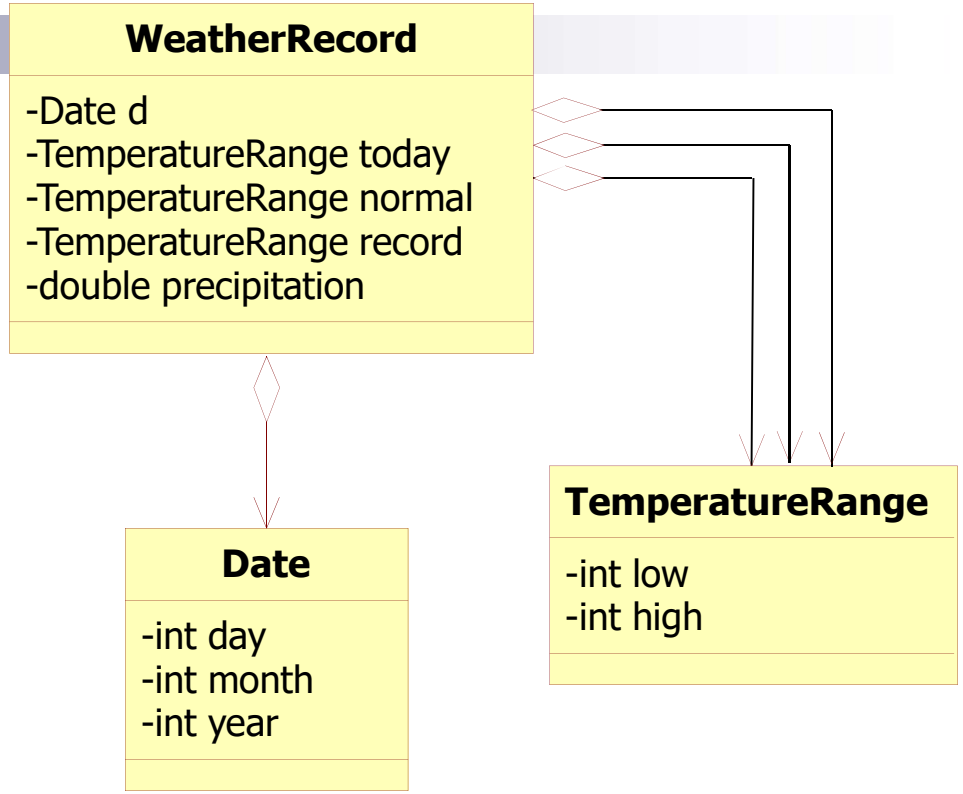
Solution

Extended exercise 3.2

- Extending exercise 3.2 with:
  - A author has a name and birth date (include year, month, day)
  - Design `equals()` method for `Author` to use in `thisAuthor()` and `sameAuthor()` methods

Exercise 3.3

- Provides the data definition for a weather recording program.



- Develop the following methods:
  - **`withinRange`**, which determines whether today's high and low were within the normal range;
  - **`rainyDay`**, which determines whether the precipitation is higher than some given value;
  - **`recordDay`**, which determines whether the temperature today broke either the high or the low record.

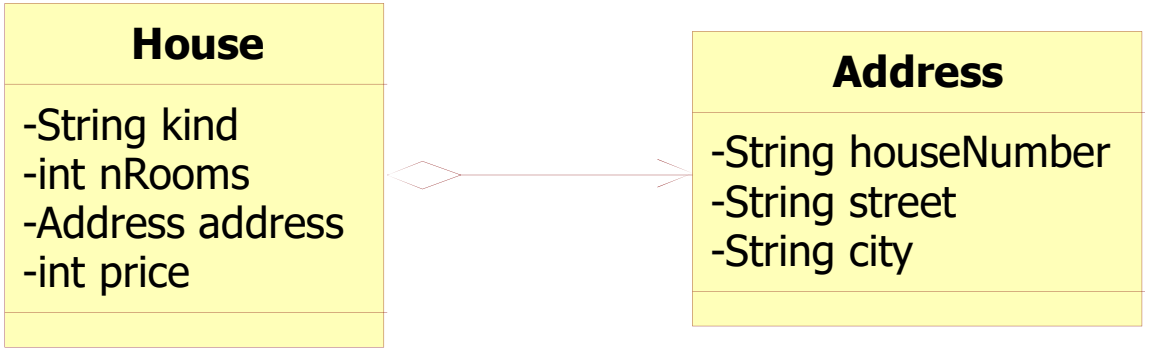
Solution

Exercises 3.4 (Lab hours)

- Develop a program that can assist railway travelers with the arrangement of train trips.
- The available information about a specific train includes its schedule, its route, and whether it is local.
- The route information consists of the origin and the destination station.
- A schedule specifies the departure and the arrival (clock) times when the train leaves and when it arrives.
- ClockTime consists of the hour (of the day) and the minutes (of the hour).
- The customer want to know:
  - Does his destination station match the destination of the train trip?
  - What time does the train start ?
  - How long does the train trip take?

Solution

Solution 3.1.3



Solution 3.1: House class

```
public class House {
   private String kind;
   private int nRooms;
   private Address address;
   private int price;

   public House(String kind, int nRooms,
                Address address, int price) {
      this.kind = kind;
      this.nRooms = nRooms;
      this.address = address;
      this.price = price;
   }
   ...
}
```

Solution 3.1: `Address` class

```
public class Address {
   private String houseNumber;
   private String street;
   private String city;

   public Address(String houseNumber, String street,
                  String city) {
      this.houseNumber = houseNumber;
      this.street = street;
      this.city = city;
   }
   ...
}
```

hasMoreRooms() and unit testing

- Method implementation

```
// class House
public boolean hasMoreRooms(House that) {
   return this.nRooms > that.nRooms;
}
```

- Unit Test

```
pubblic class HouseTest extends TestCase {
   public void testHasMoreRooms() {
      House house1 = new House("Ranch", 7,
        new Address("23", "Mapple Street", "Brooklyn"), 375000);
      House house2 = new House("Colonial", 9,
        new Address("5", "Jove Road", "Newton"), 450000);
      House house3 = new House("Cape", 6,
        new Address("83", "Winslow Road", "Waltham"), 235000);
      assertFalse(house1.hasMoreRooms(house2));
      assertTrue(house2.hasMoreRooms(house3));
   }
}
```

inThisCity()

```
// class House
public boolean inThisCity(String city) {
   return this.address.inThisCity(city);
}
```

```
// class Address
public boolean inThisCity(String thatCity) {
   return this.city.equals(thatCity);
}
```

`inThisCity()` unit testing

```
public void testThisCity() {
   House house1 = new House("Ranch", 7,
       new Address("23", "Mapple Street", "Brooklyn"), 375000);
   House house2 = new House("Colonial", 9,
       new Address("5", "Jove Road", "Newton"), 450000);
   House house3 = new House("Cape", 6,
       new Address("83", "Winslow Road", "Waltham"), 235000);
   assertTrue(house1.inThisCity("Brooklyn"));
   assertFalse(house1.inThisCity("Newton"));
}
```

sameCity()

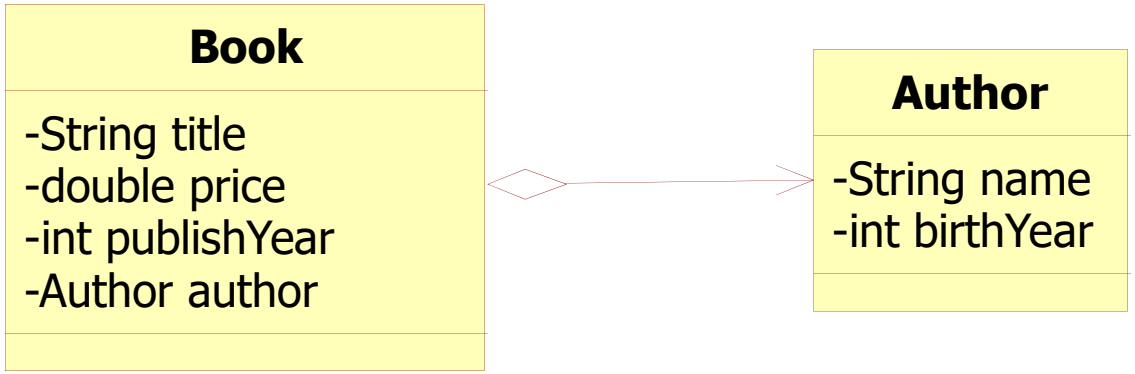
```
// class House
public boolean sameCity(House that) {
   return this.address.sameCity(that.address);
}
```

```
// class Address
public boolean sameCity(Address that) {
   return this.city.equals(that.city);
}
```

`sameCity()` unit testing

```
public void testSameCity() {
   House house1 = new House("Ranch", 7,
       new Address("23", "Mapple Street", "Brooklyn"), 375000);
   House house2 = new House("Colonial", 9,
          new Address("5", "Jove Road", "Newton"), 450000);
   House house3 = new House("Cape", 6,
          new Address("83", "Winslow Road", "Waltham"), 235000);
   assertTrue(house1.sameCity(house1));
   assertFalse(house1.sameCity(house2));
}
```

Solution 3.2


Book
-String title
-double price
-int publishYear
-Author author
Author
-String name
-int birthYear

currentBook()

- Method implementation

```
public boolean currentBook() {
   return (this.publishYear == 2004) ||
          (this.publishYear == 2003);
}
```

currentBook() unit testing

```
public void testCurrentBook() {
   Author felleisen =
         new Author("Matthias Felleisen", 1960);
   Book htdch = new Book(felleisen,
         "How to Design Class Hierarchies", 0.0, 2004);
   assertTrue(htdch.currentBook());
   Author friedman = new Author("Daniel P. Friedman", 1939);
   Book aljafp = new Book(friedman,
         "A Little Java, A Few Pattern", 25.9, 1998);
   assertFalse(aljafp.currentBook());
}
```

currentAuthor()

- Method implementation

```
// in class Book
public boolean currentAuthor() {
   return this.author.currentAuthor();
}
```

```
// in class Author
public boolean currentAuthor() {
   return this.birthYear >= 1940;
}
```

`currentAuthor()` unit testing

```
public void testCurrentAuthor() {
   Author felleisen = new Author("Matthias Felleisen", 1960);
   Book htdch = new Book(felleisen,
            "How to Design Class Hierarchies", 0.0, 2004);
   assertTrue(htdch.currentAuthor());

   Author friedman = new Author("Daniel P. Friedman", 1939);
   Book aljafp = new Book(friedman,
            "A Little Java, A Few Pattern", 25.9, 1998);
   assertFalse(aljafp.currentAuthor());
}
```

thisAuthor()

- Method implementation

```
// in class Book
public boolean thisAuthor(Author that) {
   return this.author.same(that);
}
```

```
// in class Author
public boolean same(Author that) {
   return (this.name.equals(that.name)) &&
          (this.birthYear == that.birthYear);
}
```

`thisAuthor()` unit testing

```
public void testThisAuthor() {
   Author felleisen = new Author("Matthias Felleisen", 1960);
   Book htdch = new Book(felleisen,
          "How to Design Class Hierarchies", 0.0, 2004);
   assertTrue(htdch.thisAuthor(felleisen));

   Author friedman = new Author("Daniel P. Friedman", 1939);
   assertFalse(htdch.thisAuthor(friedman));
}
```

sameAuthor()

- Method implementation

```
// in class Book
public boolean sameAuthor(Book that) {
   return this.author.same(that.author);
}
```

`sameAuthor()` unit testing

- Unit testing

```
public void testSameAuthor() {
   Author felleisen = new Author("Matthias Felleisen", 1960);
   Book htdch = new Book(felleisen,
         "How to Design Class Hierarchies", 0.0, 2004);
   Book htdp = new Book(felleisen,
         "How to Design Programs", 0.0, 2002);
   assertTrue(htdch.sameAuthor(htdp));

   Author friedman = new Author("Daniel P. Friedman", 1939);
   Book aljafp = new Book(friedman,
         "A Little Java, A Few Pattern", 25.9, 1998);
   assertFalse(aljafp.sameAuthor(htdch));
}
```

sameGeneration()

- Method implementation

```
// in class Book
public boolean sameGeneration(Book that) {
   return this.author.sameGeneration(that.author);
}
```

```
// in class Author
public boolean sameGeneration(Author that) {
   return Math.abs(this.birthYear - that.birthYear) <= 10;
}
```

sameGeneration() unit testing

```
public void testSameGeneration() {
   Author felleisen = new Author("Matthias Felleisen", 1960);
   Book htdch = new Book(felleisen,
         "How to Design Class Hierarchies", 0.0, 2004);
   assertTrue(htdch.sameGeneration(htdp));

   Author friedman = new Author("Daniel P. Friedman", 1939);
   Book aljafp = new Book(friedman,
         "A Little Java, A Few Pattern", 25.9, 1998);
   assertFalse(aljafp.sameGeneration(htdch));
}
```

Back

Solution 3.3

WeaherRecord class definition

```
public class WeatherRecord {
   private Date d;
   private TemperatureRange today;
   private TemperatureRange normal;
   private TemperatureRange record;
   private double precipitation;
   public WeatherRecord(Date d, TemperatureRange today,
                        TemperatureRange normal,
                        TemperatureRange record,
                        double precipitation) {
      this.d = d;
      this.today = today;
      this.normal = normal;
      this.record = record;
      this.precipitation = precipitation;
   }
}
```

WeaherRecord class definition

```
public class TemperatureRange {
   private int low;
   private int high;
   public TemperatureRange(int low, int high) {
      this.low = low;
      this.high = high;
   }
}
```

```
public class Date {
   private int day;
   private int month;
   private int year;
   public Date(int day, int month, int year) {
      this.day = day;
      this.month = month;
      this.year = year;
   }
}
```

withinRange()

```
public class WeatherRecord {
   ...

   public boolean withinRange() {
      return this.today.within(this.normal);
   }
}
```

```
public class TemperatureRange {
   private double low;
   private double high;
   public boolean within(TemperatureRange that) {
      return (this.low >= that.low) &&
             (this.high <= that.high);
   }
}
```

rainyDay()

```
public class WeatherRecord {
   ...

   public boolean rainyDay(double thatPrecipitation) {
      return this.precipitation >= thatPrecipitation;
   }
}
```

recordDay()

```
public class WeatherRecord {
   ...

   public boolean recordDay() {
      return !this.today.within(this.record);
   }
}
```

Solution 3.4

Class Diagram