Structural Programming and Data Structures

Winter 2000

CMPUT 102: Vectors and other Repetitions

Dr. Osmar R. Zaïane

University of Alberta

Course Content

Objectives of Lecture 17

Vectors and For Statements

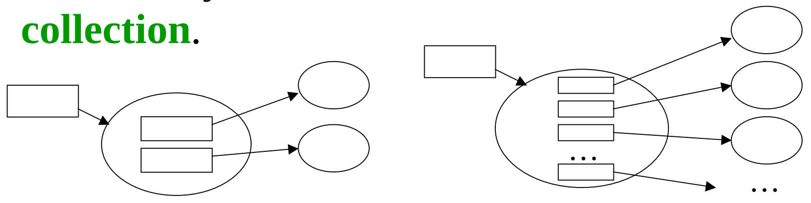
- learn about container objects that can contain an arbitrary number of other objects.
- See the Vector object as an example of a container.
- Introduce a repetition control structure called the for statement that allows the iteration over indexed collections.
- Re-write the Adventure program using the concept of vectors .

Outline of Lecture 17

- Containers
- Vectors
- The for statement
- Adventure Version 8

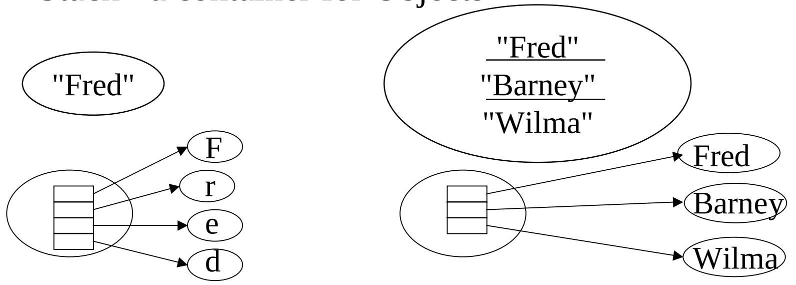
Containers

- An object's state consists of instance variables that are bound to other objects or values.
- Sometimes it is useful for an object's state to include an arbitrary number of other objects.
- An object that remembers an arbitrary number of other objects is called a **container** or a **collection**.



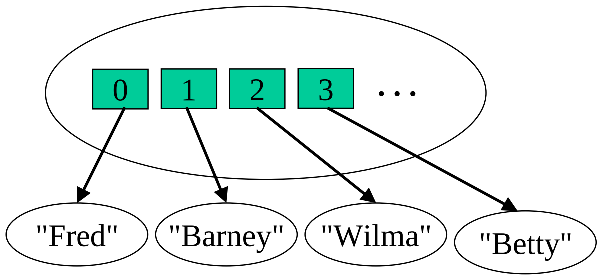
Strings and Stacks

- We have already seen two containers:
  - String - a container for characters
  - Stack - a container for Objects



Indexed Containers

- Containers whose elements are indexed by integers are called indexed containers.
- The integer indexes are the object references.



Examples of Indexed Lists

List of students and their grades

1	Jane Doe	90
2	Bob Smith	85
3	John Flint	83
4	Wilma Stone	79
5	Fred Ming	75
…		

List of cities I visited

1	Edmonton
2	Vancouver
3	Denver
4	San Diego

Who is the 10th student?

What is the 3 city?

With a stack, I can only see the top element.

Outline of Lecture 17

- Containers
- Vectors
- The for statement
- Adventure Version 8

Java - Vector 1

- Java has a class called Vector, that is an ordered indexed container of an arbitrary number of Objects.
- A Vector, can hold any kind of Objects, but not values.

```
Vector        myCities;
myCities = new Vector();
myCities.addElement("Edmonton");
myCities.addElement("Vancouver");
myCities.addElement("Denver");
myCities.addElement("San Diego");
```

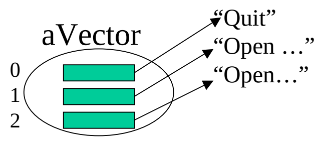
Vector addElement

Java - Vector 2

- A Vector is indexed by non-negative ints so it can be accessed by position
- The first position is 0, not 1.
- A Vector knows its current size.
- A Vector can be iterated by index.
- When you access an Object in a Vector, you often must **cast** its type to use it.

Java - Vector 3

```
String  element;
int index;
Vector aVector;
aVector = new Vector();
aVector.addElement("Quit");
aVector.addElement("Open the chest");
aVector.addElement("Open the blue door");

index = 0;
While (index < aVector.size()) {
   element = (String) aVector.elementAt(index);
   System.out.println(element);
   index = index + 1;
}
```



Outline of Lecture 17

- Containers
- Vectors
- The for statement
- Adventure Version 8

Java Syntax: for Statement

- A **for** statement is a special repetition control structure for indexed collections.
- The syntax of a for statement in Java is:

<for statement> ::= for (<assignment>; <condition>; <increment>)
    <statement>

- For example:

```
for (index = 0; index < aVector.size(); index++) {
     element = (String) aVector.elementAt(index);
     System.out.println(element);
}
```

Java shorthand for:
index = index + 1;

Semantics - for

- The assignment is executed to initialize the index variable.
- The condition is evaluated and if it is true the statement is executed.
- The increment is performed on the index variable.
- The condition is re-evaluated and if it is true, the statement is executed again.
- This continues until the condition is false at which time the for statement is done.

for Semantics - Example

```
Vector aVector;
int    index;
String element;
aVector = new Vector();
aVector.addElement("Quit");
aVector.addElement("Open the chest");
aVector.addElement("Open the blue door");
for (index = 0; index < aVector.size(); index++) {
     element = (String) aVector.elementAt(index);
     System.out.println(element);
}
```

```
Quit
Open the Chest
Open the blue door
```

Outline of Lecture 17

- Containers
- Vectors
- The for statement
- Adventure Version 8

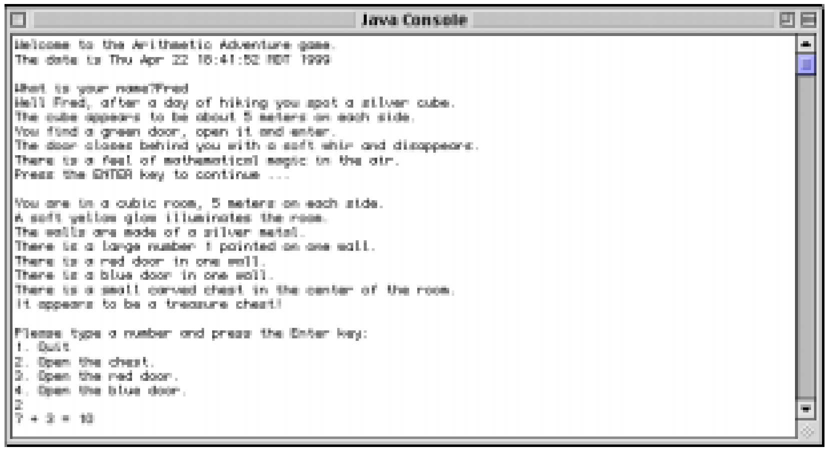
Adventure 8

- Use Vectors to modify the Arithmetic Adventure game so that many rooms are supported.
- Use Vectors to improve the implementation of TextMenu.

Adventure 8 - Changes Summary

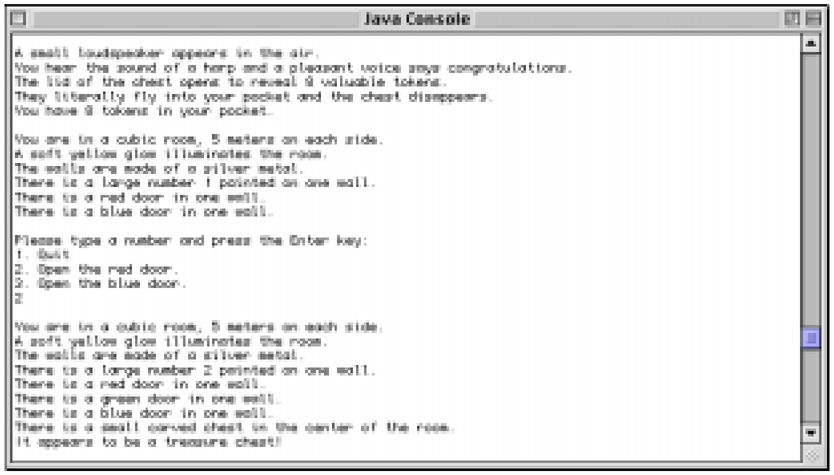
- Add the class Door.
- Make many changes to class Adventure.
- Make many changes to class Room.
- Make changes to class TextMenu.
- Leave the classes: Adventurer, RandomInt, Chest and Question unchanged.

Running Adventure 8 (1)

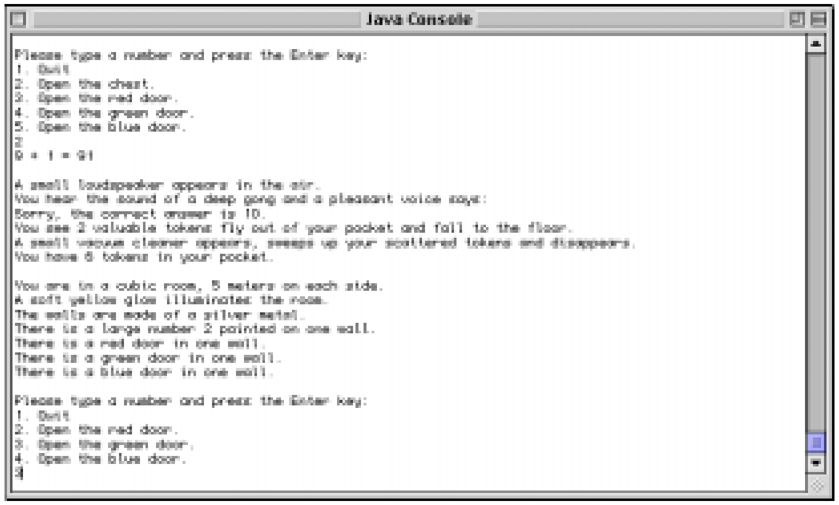


```
Welcome to the Arithmetic Adventure game.
The date is Thu Apr 22 16:41:52 MDT 1999

What is your name?Fred
Well Fred, after a day of hiking you spot a silver cube.
The cube appears to be about 5 meters on each side.
You find a green door, open it and enter.
The door closes behind you with a soft whir and disappears.
There is a feel of mathematical magic in the air.
Press the ENTER key to continue ...

You are in a cubic room, 5 meters on each side.
A soft yellow glow illuminates the room.
The walls are made of a silver metal.
There is a large number 1 painted on one wall.
There is a red door in one wall.
There is a blue door in one wall.
There is a small carved chest in the center of the room.
It appears to be a treasure chest!

Please type a number and press the Enter key:
1. Quit
2. Open the chest.
3. Open the red door.
4. Open the blue door.
2
7 + 3 = 10
```

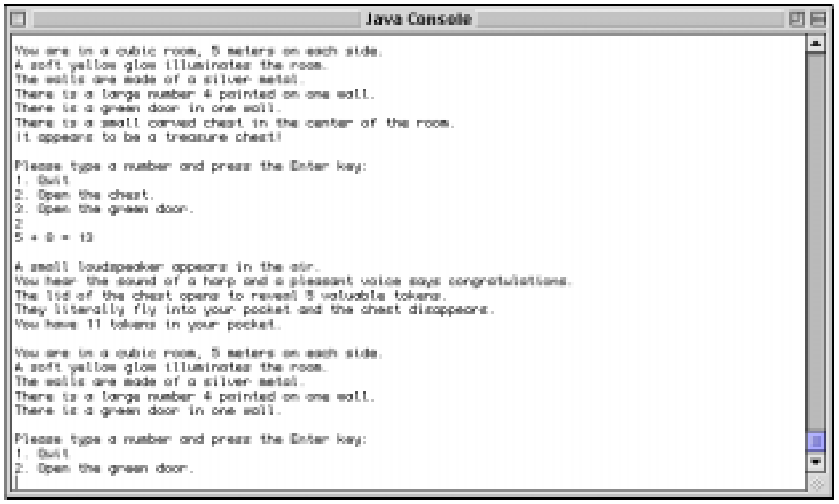
Running Adventure 8 (2)


```
A small loudspeaker appears in the air.
You hear the sound of a harp and a pleasant voice says congratulations.
The lid of the chest opens to reveal 9 valuable tokens.
They literally fly into your pocket and the chest disappears.
You have 9 tokens in your pocket.

You are in a cubic room, 5 meters on each side.
A soft yellow glow illuminates the room.
The walls are made of a silver metal.
There is a large number 1 painted on one wall.
There is a red door in one wall.
There is a blue door in one wall.

Please type a number and press the Enter key:
1. Quit
2. Open the red door.
3. Open the blue door.
2

You are in a cubic room, 5 meters on each side.
A soft yellow glow illuminates the room.
The walls are made of a silver metal.
There is a large number 2 painted on one wall.
There is a red door in one wall.
There is a green door in one wall.
There is a blue door in one wall.
There is a small carved chest in the center of the room.
It appears to be a treasure chest!
```

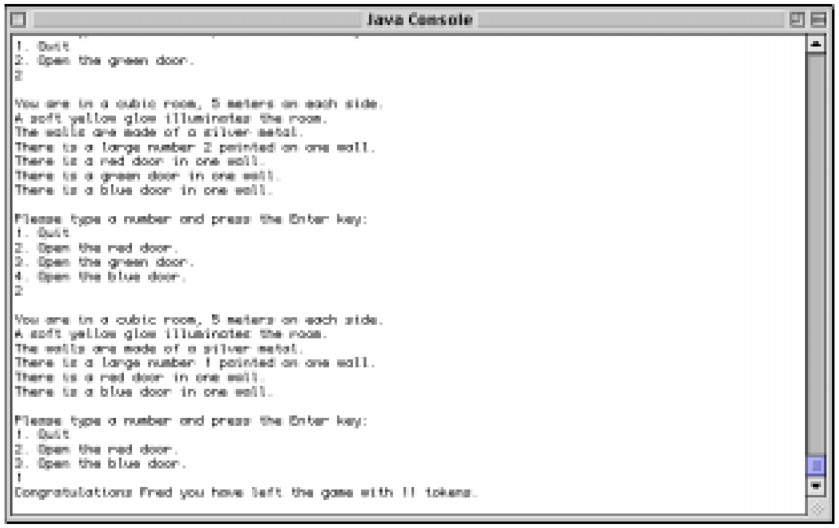
Running Adventure 8 (3)


```
Please type a number and press the Enter key:
1. Quit
2. Open the chest.
3. Open the red door.
4. Open the green door.
5. Open the blue door.
2
9 + 1 = 91

A small loudspeaker appears in the air.
You hear the sound of a deep gong and a pleasant voice says:
Sorry, the correct answer is 10.
You see 3 valuable tokens fly out of your pocket and fall to the floor.
A small vacuum cleaner appears, sweeps up your scattered tokens and disappears.
You have 6 tokens in your pocket.

You are in a cubic room, 5 meters on each side.
A soft yellow glow illuminates the room.
The walls are made of a silver metal.
There is a large number 2 painted on one wall.
There is a red door in one wall.
There is a green door in one wall.
There is a blue door in one wall.

Please type a number and press the Enter key:
1. Quit
2. Open the red door.
3. Open the green door.
4. Open the blue door.
3
```

Running Adventure 8 (4)


```
You are in a cubic room, 5 meters on each side.
A soft yellow glow illuminates the room.
The walls are made of a silver metal.
There is a large number 4 painted on one wall.
There is a green door in one wall.
There is a small carved chest in the center of the room.
It appears to be a treasure chest!

Please type a number and press the Enter key:
1. Quit
2. Open the chest.
3. Open the green door.
2
5 + 8 = 13

A small loudspeaker appears in the air.
You hear the sound of a harp and a pleasant voice says congratulations.
The lid of the chest opens to reveal 5 valuable tokens.
They literally fly into your pocket and the chest disappears.
You have 11 tokens in your pocket.

You are in a cubic room, 5 meters on each side.
A soft yellow glow illuminates the room.
The walls are made of a silver metal.
There is a large number 4 painted on one wall.
There is a green door in one wall.

Please type a number and press the Enter key:
1. Quit
2. Open the green door.
```

Running Adventure 8 (5)


```
1. Quit
2. Open the green door.
2

You are in a cubic room, 5 meters on each side.
A soft yellow glow illuminates the room.
The walls are made of a silver metal.
There is a large number 2 painted on one wall.
There is a red door in one wall.
There is a green door in one wall.
There is a blue door in one wall.

Please type a number and press the Enter key:
1. Quit
2. Open the red door.
3. Open the green door.
4. Open the blue door.
2

You are in a cubic room, 5 meters on each side.
A soft yellow glow illuminates the room.
The walls are made of a silver metal.
There is a large number 1 painted on one wall.
There is a red door in one wall.
There is a blue door in one wall.

Please type a number and press the Enter key:
1. Quit
2. Open the red door.
3. Open the blue door.
1
Congratulations Fred you have left the game with 11 tokens.
```

Program - Adventure 7.1

OLD

```
import java.util.*;
public class Adventure {
/* Version 7
    This program is an arithmetic adventure game …
*/

/* Constructors */

    public Adventure () {
    /*
            Initialize an adventure by creating the appropriate
            objects.
    */
            this.firstRoom = new Room(1);
    }
```

Program - Adventure 8.1

NEW

```
import java.util.*;
public class Adventure {
/* Version 8 This program is an arithmetic … */
/* Constructors */
    public Adventure () {
    /*    Initialize an adventure by creating the appropriate
          objects.
    */
          Vector rooms;
          int i;
          rooms = new Vector();
          for (i = 0; i <= 4; i++)
                    rooms.addElement(new Room(i + 1));
          this.makeDoor(rooms, 1, 2, "red");
```

OLD

Program - Adventure 7.2

```
/* Main program */

    public static void main(String args[]) {
            Adventure       game;

            game = new Adventure();
            game.play();
    }

/* Private Instance Variables */

    private Room firstRoom;
```

Program - Adventure 8.2

NEW

```
          this.makeDoor(rooms, 1, 3, "blue");
          this.makeDoor(rooms, 2, 4, "green");
          this.makeDoor(rooms, 2, 5, "blue");
          this.firstRoom = (Room) rooms.elementAt(0);
    }

/* Main program */
    public static void main(String args[]) {
          Adventure       game;
          game = new Adventure();
          game.play();
    }
/* Private Instance Variables */
    private Room firstRoom;
```

Program - Adventure 7.3

OLD

```
/* Private Instance Methods */

   private void play() {
   /*
        Plays the Adventure game.
   */

        Adventurer     adventurer;
        Room           room;

        adventurer = this.greeting();
        room = firstRoom.enter(adventurer);
        this.farewell(adventurer);
   }
```

Program - Adventure 8.3

NEW

```
/* Private Instance Methods */
   private void play() {
   /*
        Plays the Adventure game.
   */
        Adventurer     adventurer;
        Room           room;

        adventurer = this.greeting();
        room = firstRoom;
        while (room != null)
                room = room.enter(adventurer);
        this.farewell(adventurer);
   }
```

Program - Adventure 8.4

NO CHANGES

```
   private Adventurer greeting() {
   /*
        Great the user and answer an Adventurer that
        represents the user.
   */
        String  playerName;

        System.out.println("Welcome to the Arithmetic Adventure game.");
        System.out.print("The date is ");
        System.out.println(new Date());
        System.out.println();
        System.out.print("What is your name?");
        playerName = Keyboard.in.readString();
```

NO CHANGES

Program - Adventure 8.5

```
        System.out.print("Well ");
        System.out.print(playerName);
        System.out.println(", after a day of hiking you spot a silver cube.");
        System.out.println("The cube appears to be about 5 meters on each side.");
        System.out.println("You find a green door, open it and enter.");
        System.out.println("The door closes behind you with a soft whir and disappears.");
        System.out.println("There is a feel of mathematical magic in the air.");
        Keyboard.in.pause();
        return new Adventurer(playerName);
   }
```

NO CHANGES

Program - Adventure 8.6

```
private void enterRoom(Adventurer adventurer) {
    /*
        The given adventurer has entered the
        first room.
    */
        Chest           chest;

        chest = new Chest();
        chest.display();
        chest.open(adventurer);
    }
```

NO CHANGES

Program - Adventure 8.7

```
private void farewell(Adventurer adventurer) {
/*
    Say farewell to the user and report the game result.
*/

    System.out.print("Congratulations ");
    System.out.print(adventurer.name());
    System.out.print(" you have left the game with ");
    System.out.print(adventurer.tokens());
    System.out.println(" tokens.");
}
```

NEW

Program - Adventure 8.8

```
private void makeDoor(Vector myRooms, int from, int to,
                                        String color) {
/*
    Make a Door from the Room with the given room number
    to the Room with the given room number in the given
    Vector of rooms. Use the given Door color.
*/
    Room fromRoom;
    Room toRoom;

    fromRoom = (Room) myRooms.elementAt(from - 1);
    toRoom = (Room) myRooms.elementAt(to - 1);
    fromRoom.makeDoor(toRoom, color);
}
```

OLD

Class - Room 7.1

```
import java.util.*;
public class Room {
/*
   A room contains a treasure chest and some doors to adjoining rooms.
*/
/* Constructor */
   public Room(int anInt) {
   /*
      Initialize me so that I have the given room number,
      contain a treasure chest, and no doors.
   */
      this.number = anInt;
      this.chest = new Chest();
   }
```

Class - Room 8.1

NEW

```
import java.util.*;
public class Room {
/* A room contains a treasure chest and some doors to adjoining rooms.
   */
/* Constructor */
   public Room(int anInt) {
   /*
        Initialize me so that I have the given room number,
        contain a treasure chest, and no doors.
   */
        this.number = anInt;
        this.chest = new Chest();
        this.doors = new Vector();
   }
```

Class - Room 8.2

NO CHANGES

```
/* Instance Methods */
   public Room enter(Adventurer adventurer) {
   /*Describe myself, display a list of options, and
        perform the selected option. If the user selected
        quit then return null. If the user selected to go
        to another Room then return that Room. Otherwise
        return this Room. */
        TextMenu      menu;
        String  action;
        this.display();
        menu = this.buildMenu();
        action = menu.launch();
        return this.performAction(action, adventurer);
   }
```

Class - Room 8.2a

NEW

```
   public void makeDoor(Room aRoom, String color) {
   /*
        Make a door of the given color and place it between
        me and the given Room.
   */
        Door door;

        door = new Door(color, this, aRoom);
        this.doors.addElement(door);
        aRoom.doors.addElement(door);
   }
```

Class - Room 7.3

OLD

```
/* Private Instance Variables */
   private Chest          chest;
   private int   number;
/* Private Instance Methods */
   private void display() {
   /*
        Output a description of myself.
   */
        this.displayBasic();
        this.displayDoors();
        if (this.chest != null)
                this.chest.display();
   }
```

Class - Room 8.3

NEW

```
/* Private Instance Variables */
   private Chest        chest;
   private int   number;
   private Vector       doors;
/* Private Instance Methods */
   private void display() {
   /*
        Output a description of myself.
   */
        this.displayBasic();
        this.displayDoors();
        if (this.chest != null)
                this.chest.display();
   }
```

Class - Room 8.4

NO CHANGES

```
private void displayBasic() {
/*
     Output a basic description of myself that is
     independent of my doors and contents.
*/
     System.out.println();
     System.out.println("You are in a cubic room, 5 meters on each side.");
     System.out.println("A soft yellow glow illuminates the room.");
     System.out.println("The walls are made of a silver metal.");
     System.out.print("There is a large number ");
     System.out.print(this.number);
     System.out.println(" painted on one wall.");
}
```

OLD

~~Class - Room 7.5~~

```
~~private void displayDoors() {
/*
     Output a description of all of my doors.
*/
}~~
```

Class - Room 8.5

NEW

```
private void displayDoors() {
/*
     Output a description of all of my doors.
*/
     Door    door;
     int     index;

     for (index = 0; index < this.doors.size(); index++){
             door = (Door) this.doors.elementAt(index);
             door.display();
     }
}
```

Class - Room 7.6

OLD

```
private TextMenu buildMenu() {
/*
      Create and answer a TextMenu containing the user's
      valid actions.
*/
      TextMenu       menu;

      menu = new TextMenu();
      menu.add("Quit");
      if (this.chest != null)
            menu.add("Open the chest.");
      // Add door choices here
      return menu;
}
```

Class - Room 8.6

NEW

```
private TextMenu buildMenu() {
/*    Create and answer a TextMenu containing the user's
      valid actions. */
      TextMenu             menu;
      int                  index;
      Door                 door;
      menu = new TextMenu();
      menu.add("Quit");
      if (this.chest != null)
        menu.add("Open the chest.");
      for (index = 0; index < this.doors.size(); index++){
        door = (Door) this.doors.elementAt(index);
        menu.add("Open the " + door.color() + " door.");
      }
      return menu;
}
```

Class - Room 7.7

OLD

```
private Room performAction(String action, Adventurer adventurer) {
/*    Perform the action described by the given String for
      the given Adventurer. Return the room the user
      selected, null if the user selected quit and this
      room if the user selected to open the chest. */
      if (action.equals("Open the chest.")) {
            this.chest.open(adventurer);
            this.chest = null;
            return this;
      }
      if (action.equals("Quit"))
            return null;
      return null;
}
}
```

Class - Room 8.7

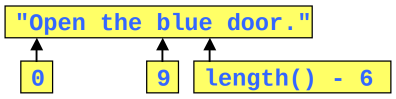
NEW

```
private Room performAction(String action, Adventurer adventurer) {
/*    Perform the action described by the given String for
      the given Adventurer. Return the room the user
      selected, null if the user selected quit and this
      room if the user selected to open the chest.
*/
      if (action.equals("Open the chest.")) {
                this.chest.open(adventurer);
                this.chest = null;
                return this;
      }
      if (action.equals("Quit"))
                return null;
      return this.getRoomForAction(action);
      }
}
```

Class - Room 8.8

NEW

```
private Room getRoomForAction(String action) {
/*      Return the Room that I am connected to that is
        represented by the given action String. If no such
        Door exists, return me.
*/
        int             index;
        String   color;
        Door            door;
        color = action.substring(9, action.length() - 6);
        for (index = 0; index < this.doors.size(); index++){
                door = (Door) this.doors.elementAt(index);
                if (color.equals(door.color()))
                        return door.adjoiningRoom(this);
        }
        return this;
}
```



Class - Door 8.1

```
public class Door {
/*
   An instance of this class represents a door in the Adventure game. A
   Door has a color and it connects two rooms.
*/
/* Constructor */
   public Door(String color, Room aRoom, Room bRoom) {
   /*
        Initialize me so that I have the given color and
        connect the given two Rooms.
   */
        this.color = color;
        this.room1 = aRoom;
        this.room2 = bRoom;
   }
```

Class - Door 8.2

```
/* Instance Methods */
   public void display() {
   /* Output a description of myself. */
        System.out.print("There is a ");
        System.out.print(this.color);
        System.out.println(" door in one wall.");
   }

   public String color() {
   /*
        Answer a String representing my color.
   */
        return this.color;
   }
```

Class - Door 8.3

```
   public Room adjoiningRoom(Room aRoom) {
   /*
        Answer the room that I connect the given Room to, or
        null if I don't connect it to any Room.
   */

        if (this.room1 == aRoom)
                return this.room2;
        else if (this.room2 == aRoom)
                return this.room1;
        else
                return null;
   }
```

Class - Door 8.4

```
/* Private Instance Variables */

    private String        color;
    private Room          room1;
    private Room          room2;

}
```

Class - TextMenu 7.1

OLD

```
import java.io.*;
import java.util.*;
public class TextMenu {
/* An instance of this class displays a list of strings for the user and
    allows the user to pick one. For now, up to
    five entries are supported. */
/* Contructor */

    public TextMenu() {
    /*
          Initialize me with no entries.
    */
          this.size = 0;
    }
```

Class - TextMenu 8.1

NEW

```
import java.io.*;
import java.util.*;
public class TextMenu {
/*
    An instance of this class displays a list of strings for the user and
    allows the user to pick one.
*/
/* Constructor */

    public TextMenu() {
    /*
          Initialize me with no entries.
    */
          this.entries = new Vector();
    }
```

Class - TextMenu 7.2

OLD

```
/* Instance Methods */
    public void add(String entry) {
    /*    Add the given String to me as my next choice. */
          this.size = this.size + 1;
          if (entry1 == null) {
                this.entry1 = entry;
                return;
          }
          if (entry2 == null) {
                this.entry2 = entry;
                return;
          }
    //more of the same for entries 3, 4 and 5.
    }
```

NEW

Class - TextMenu 8.2

```
/* Instance Methods */
   public void add(String entry) {
   /*
         Add the given String to me as my next choice.
   */
         this.entries.addElement(entry);
   }
```

OLD

Class - TextMenu 7.3

```
public String launch() {
/*
      Display myself and answer the String entry selected
      by the user.
*/
      String  action;
      int              index;

      index = this.getUserSelection();
```

DELETED

Class - TextMenu 7.4

```
      switch (index) {
            case 1: action = this.entry1; break;
            case 2: action = this.entry2; break;
            case 3: action = this.entry3; break;
            case 4: action = this.entry4; break;
            case 5: action = this.entry5; break;
            default: action = "";
      }
      return action;
   }
```

NEW

Class - TextMenu 8.3

```
public String launch() {
/*
      Display myself and answer the String entry selected
      by the user.
*/
      String  action;
      int              index;

      index = this.getUserSelection();
      action = (String) this.entries.elementAt(index - 1);
      return action;
}
```

OLD

Class - TextMenu 7.5

```
/* Private Instance Variables */

    private String entry1;
    private String entry2;
    private String entry3;
    private String entry4;
    private String entry5;
    private int size;

/* Private Instance Methods */
```

NEW

Class - TextMenu 8.5

```
/* Private Instance Variables */

    private Vector entries;

/* Private Instance Methods */
```

Class - TextMenu 7.6

OLD

```
    private void display() {
    /*
        Display myself on the screen.
    */
        String  entry;
        int                 index;
        System.out.println();
        System.out.println("Please type a number and press the Enter key:");
        if (this.entry1 != null) {
                System.out.print("1. ");
                System.out.println(this.entry1);
        }
        // same code for entry2, entry3, entry4 and entry5
    }
}
```

Class - TextMenu 8.6

NEW

```
private void display() {
/*
    Display myself on the screen.
*/
  String       entry;
  int index;
  System.out.println();
  System.out.println("Please type a number and press the Enter key:");
  for (index = 0; index < this.entries.size(); index++){
      entry = (String) this.entries.elementAt(index);
      System.out.print(index + 1);
      System.out.print(". ");;
      System.out.println(entry);
  }
}
```

Class - TextMenu 8.7

NO CHANGE

```
  private int getUserSelection() {
/*
    Query the user for an action and answer the index of
    the choice. If the user does not answer with a valid
    action, query again.
*/

    Integer choice;
    int         index;

    index = 0;
```

OLD

~~Class - TextMenu 7.8~~

~~while ((index < 1) || (index > this.size)) {~~
~~this.display();~~
~~choice = Keyboard.in.readInteger();~~
~~if (choice == null)~~
~~index = 0;~~
~~else~~
~~index = choice.intValue();~~
~~}~~
~~return index;~~

Class - TextMenu 8.8

NEW

```
   while ((index < 1)||(index > this.entries.size())) {
    this.display();
    choice = Keyboard.in.readInteger();
    if (choice == null)
          index = 0;
    else
          index = choice.intValue();
  }
  return index;
 }
}
```

The Rest of the Classes are Omitted

- The rest of the classes are omitted to save space.
- See Lecture 16 and 15 for missing classes.