enum Types, Audio & Graphics, TicTacToe Game v1.0

Dr. 何明昕, He Mingxin, Max

program06 @ yeah.net

Email Subject：(AE|A2|A3) + *(Last 4 digits of ID)* + Name: *TOPIC*

Sakai: CS102A in 2018A

计算机程序设计基础

Introduction to Computer Programming

enum Types

Ch08, 8.9, Java™ How to Program, 10th ed.

More IO: Audio & Graphics

TextBook: 1.5 Input and Output

TextBook: 2.1 Static Methods
PlayThatTuneDeluxe.java

OOP Case Studies:
Tic-Tac-Toe Game v1.0

Jhtp10, 8.9 enum Types - I

- The basic enum data type defines a set of constants represented as unique identifiers.
- enum types in Java are reference types (a special class type).
- An enum type may be declared with an enum declaration, which is a comma separated list of enum constants.
- The declaration may optionally include other components of traditional classes, such as constructors, fields and methods.
- The constructor for an enum type must have package or private access.
- This constructor automatically creates and initializes the constants that are defined at the beginning of the enum body.
- It is a syntax error to invoke an enum constructor in your code.

```
public enum Suit { SPADE, HEART, DIAMOND, CLUB };


Suit s1 = Suit.HEART;


if (s1 != Suit.CLUB) {
    ...
}

switch (sx) {
    case Suit.SPADE: ... break;
    case Suit.HEART:
    case Suit.DIAMOND: ... break;
    case Suit.CLUB: ... break;
}
```

8.9 enum Types - II

- Each enum declaration declares an enum class with the following restrictions:
  - enum member constants are implicitly final and cannot be modified.
  - enum constants are implicitly static.
  - Any attempt to create an object of an enum type with operator new followed by the constructor call results in a compilation error.
  - enum constants can be used anywhere constants can be used, such as in the case labels of switch statements and to control enhanced for statements.
  - enum declarations contain two parts - the enum constants and the other members of the enum type.
  - An enum constructor can specify any number of parameters and can be overloaded.

8.9 enum Types - III

- For every enum declaration, the compiler generates the static method called `values()` that returns an array of all enum's constants.
- When an enum constant is converted to a String, the constant's identifier is used as its String representation.
- ch08\fig08_10_11\Book.java
- ch08\fig08_10_11\EnumTest.java

8.9 enum Types - IV

- The static method range of class EnumSet (declared in package java.util) is used to access a range of an enum's constants.
  - Method range takes two parameters - the first and the last enum constants in the range
  - Returns an EnumSet that contains all the constants between these two constants, inclusive.
- The enhanced for statement can be used with an EnumSet just as it can with an array.
- Class EnumSet provides several other static methods.

```java
// Fig. 8.10: Book.java
// Declare an enum type with constructor and explicit instance fields
// and accessors for these fields

public enum Book {
   // declare constants of enum type
   JHTP( "Java How to Program", "2015"),
   CHTP( "C How to Program", "2013"),
   IW3HTP( "Internet & World Wide Web How to Program", "2012"),
   CPPHTP( "C++ How to Program", "2014"),
   VBHTP( "Visual Basic How to Program", "2014"),
   CSHARPHTP( "Visual C# How to Program", "2014");

   // instance fields
   private final String title;
   private final String copyrightYear;

   // enum constructor
   Book (String title, String copyrightYear) {
      this.title = title;
      this.copyrightYear = copyrightYear;
   }

   // accessor for field title
   public String getTitle () {
      return title;
   }

   // accessor for field copyrightYear
   public String getCopyrightYear () {
      return copyrightYear;
   }
}
```

```java
// Fig. 8.11: EnumTest.java
// Testing enum type Book.
import java.util.EnumSet;

public class EnumTest {
   public static void main (String[] args) {
      System.out.println( "All books:");

      // print all books in enum Book
      for (Book book : Book.values())
         System.out.printf( "%-10s%-45s%s\n",
            book, book.getTitle(), book.getCopyrightYear()
         );

      System.out.printf( "\nDisplay a range of enum constants:\n");

      // print first four books
      for (Book book : EnumSet.range( Book.JHTP, Book.CPPHTP))
         System.out.printf( "%-10s%-45s%s\n",
            book, book.getTitle(), book.getCopyrightYear()
         );
   }
}

```

```
All books:
JHTP       Java How to Program                              2015
CHTP       C How to Program                                 2013
IW3HTP     Internet & World Wide Web How to Program         2012
CPPHTP     C++ How to Program                               2014
VBHTP      Visual Basic How to Program                      2014
CSHARPHTP  Visual C# How to Program                         2014

Display a range of enum constants:
JHTP       Java How to Program                              2015
CHTP       C How to Program                                 2013
IW3HTP     Internet & World Wide Web How to Program         2012
CPPHTP     C++ How to Program                               2014
```

Fig. 8.11 | Testing `enum` type Book.

枚举的常见方法

Enum抽象类常见方法

Enum是所有 Java 语言枚举类型的公共基本类（注意Enum是抽象类），以下是它的常见方法：

返回类型	方法名称	方法说明
`int`	`compareTo(E o)`	比较此枚举与指定对象的顺序
`boolean`	`equals(Object other)`	当指定对象等于此枚举常量时，返回 true。
`Class<?>`	`getDeclaringClass()`	返回与此枚举常量的枚举类型相对应的 Class 对象
`String`	`name()`	返回此枚举常量的名称，在其枚举声明中对其进行声明
`int`	`ordinal()`	返回枚举常量的序数（它在枚举声明中的位置，其中初始常量序数为零）
`String`	`toString()`	返回枚举常量的名称，它包含在声明中
`static<T extends Enum<T>> T`	`static valueOf(Class<T> enumType, String name)`	返回带指定名称的指定枚举类型的枚举常量。

More Input & Output: Audio & Graphics

TextBook: 1.5 Input and Output

https://introcs.cs.princeton.edu/java/15inout/

TextBook: 2.1 Static Methods

PlayThatTuneDeluxe.java

https://introcs.cs.princeton.edu/java/21function/

OOP Case Studies:
Tic-Tac-Toe Game v1.0

Tool.java

Move.java

Board.java

TicTacToe.java

TicTacToeGame.java

Tool.java

```java
/**
   A class for Tool (of Player)
 */
public enum Tool {
  X, O, EMPTY;   // name()

  public String toString () { return SHOW_AS[ ordinal() ]; }

  private static String[] SHOW_AS = { "X", "O", " " };
}
```

Move.java

```java
/**
   A class for Tic Tac Toe moves
*/
public class Move {
  public Move (int r, int c) {
    if (r < 1 || Board.SIZE < r || c < 1 || Board.SIZE < c)
      throw new IllegalArgumentException();

    row = r;
    column = c;
  }

  public int getRow ()  { return row; }

  public int getColumn ()   { return column; }

  private int row;
  private int column;
}
```

Board.java

```
1  /**
2     A class for storing, manipulating, and printing TicTacToe boards
3  */
4  public class Board {
5    /*
6      A sample board is shown like this:
7
8        Here is the current board:
9
10       o| |x
11       -+-+-
12       x|o|o
13       -+-+-
14        | |x
15
16   */
```

Board.java

```
17  public void show () {
18    System.out.println( "Here is the current board:\n" );
19
20    for (int r = 1; r <= SIZE; r++) {
21      for (int c = 1; c <= SIZE; c++) {
22        System.out.print( board[r-1][c-1] );
23        if (c != SIZE)   // Print strut after all but last column
24          System.out.print( "|" );
25      }
26      System.out.println();
27
28      if (r != SIZE)   // Print row line after all but last row
29        System.out.println( "-+-+-" );
30    }
31    System.out.println();
32  }
33
```

Board.java

```
34  public boolean isGameWon () {
35    final Tool[][] b = board;  // a local variable for shorter expressions
36
37    // Check (short circuit) all rows, columns and diagonals for a win
38    return
39      b[0][0]!=Tool.EMPTY && b[0][0]==b[0][1] && b[0][1]==b[0][2] ||  // Row 0
40      b[1][0]!=Tool.EMPTY && b[1][0]==b[1][1] && b[1][1]==b[1][2] ||  // Row 1
41      b[2][0]!=Tool.EMPTY && b[2][0]==b[2][1] && b[2][1]==b[2][2] ||  // Row 2
42
43      b[0][0]!=Tool.EMPTY && b[0][0]==b[1][0] && b[1][0]==b[2][0] ||  // Col 0
44      b[0][1]!=Tool.EMPTY && b[0][1]==b[1][1] && b[1][1]==b[2][1] ||  // Col 1
45      b[0][2]!=Tool.EMPTY && b[0][2]==b[1][2] && b[1][2]==b[2][2] ||  // Col 2
46
47      b[1][1]!=Tool.EMPTY && b[0][0]==b[1][1] && b[1][1]==b[2][2] ||  // Dia 1
48      b[1][1]!=Tool.EMPTY && b[2][0]==b[1][1] && b[1][1]==b[0][2] ;   // Dia 2
49  }
50
51  public boolean isFull () {
52    for (int i = 0; i < SIZE; i++)
53      for (int j = 0; j < SIZE; j++)
54        if (board[i][j] == Tool.EMPTY) return false;
55    return true;
56  }
```

Board.java

```
57
58    public boolean isValid (Move move) {
59      int r = move.getRow();
60      int c = move.getColumn();
61      return board[r-1][c-1] == Tool.EMPTY;
62    }
63
64    public void handleMove (Move move, Tool player) {
65      int r = move.getRow();
66      int c = move.getColumn();
67      System.out.printf( "\nThe move for %s is %d, %d\n", player, r, c );
68
69      board[r-1][c-1] = player;  // Place the player's Tool on the board
70    }
71
72    public void clear () {
73      for (int i = 0; i < SIZE; i++)
74        for (int j = 0; j < SIZE; j++)
75          board[i][j] = Tool.EMPTY;
76    }
77
78    public static final int SIZE = 3;
79    private Tool[][] board = new Tool[SIZE][SIZE];
80  }
```

TicTacToe.java

```java
/**
   A class playing a game of TicTacToe
*/
import java.util.Random;
import java.util.Scanner;

public class TicTacToe {
  public void play () {
    showStartHint();
    randomFirstPlayer();

    board.clear();
    board.show();

    boolean gameOver = false;
    while (!gameOver) {
      Move move = getAMove();
      board.handleMove( move, player );

      board.show();

      if (board.isGameWon() || board.isFull())
        gameOver = true;
      else
        player = oppositePlayer();
    }

    showGameResult();
  }
```

TicTacToe.java

```
31    private void showStartHint() {
32      System.out.println( HINT_MESSAGE);
33    }
34
35    private void randomFirstPlayer () {
36      if (generator.nextBoolean()) {
37        person   = Tool.X;
38        computer = Tool.O;
39      } else {
40        person   = Tool.O;
41        computer = Tool.X;
42      }
43      player = Tool.X;
44    }
45
46    private Tool oppositePlayer () {
47      return (player == computer) ? person : computer;
48    }
49
50    private void showGameResult () {
51      if (board.isGameWon())
52        System.out.println( player==person ? "You won!" : "I won!" );
53      else if (board.isFull())
54        System.out.println( "We tied!" );
55      else
56        System.out.println( "Something went wrong!" );
57    }
```

TicTacToe.java

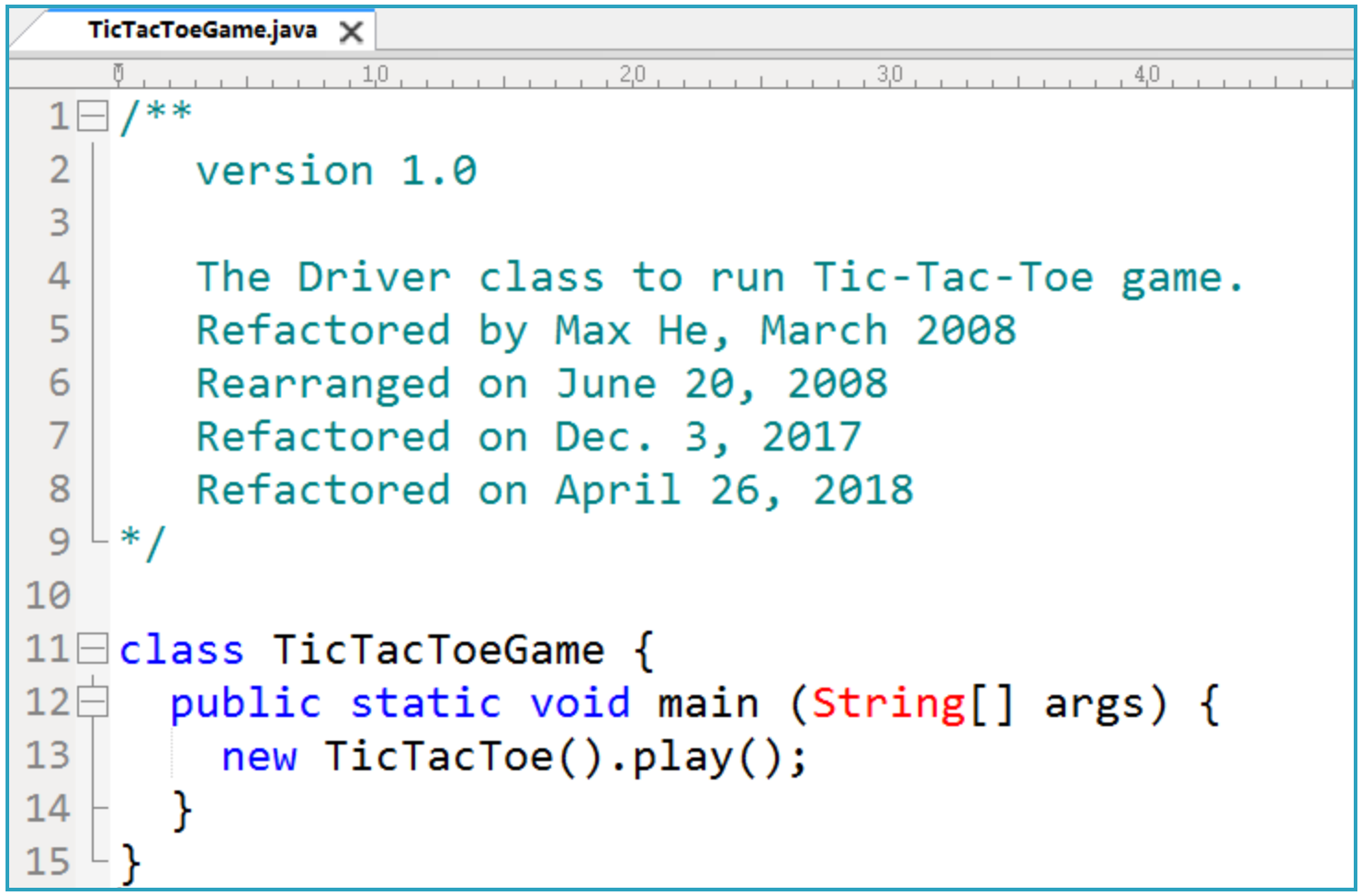
```
59    // Creates a move, either a random generated move or as input from the user
60    private Move getAMove () {
61      Move move = null;
62
63      if (player == computer) {
64        System.out.println( "It is my move.  I am '" + player + "'" );
65        move = randomGenerateAValidMove();
66      } else {
67        System.out.println( "It is your move.  You are '" + player + "'" );
68        move = getAValidMoveFromPerson();
69      }
70      return move;
71    }
72
73    private Move randomGenerateAValidMove () {
74      Move move = null;
75      do {
76        move = new Move( intBetween(1, Board.SIZE), intBetween(1, Board.SIZE));
77      } while ( !board.isValid( move ) );
78      return move;
79    }
80
```

TicTacToe.java

```
81  private int intBetween (int low, int high) {
82    return low + generator.nextInt( high - low + 1 );
83  }
84
85  private Move getAValidMoveFromPerson () {
86    Move move = null;
87    while (true) {
88      try {
89        System.out.print( "Enter a row and column on one line: " );
90        move = new Move( in.nextInt(), in.nextInt() );
91        // Generates an exception if can't make a move from r and c
92
93        if (board.isValid( move )) return move;
94
95        System.out.println( "Invalid move. Try again!" );
96      } catch (Exception e) {
97        System.out.println( "Input error. Try again!" );
98      }
99    }
100 }
101
```

TicTacToe.java

```
101
102     private Tool player;
103     private Tool computer;
104     private Tool person;
105
106     private Board board = new Board();
107     private Scanner in = new Scanner( System.in );
108
109     private static Random generator = new Random();
110
111     private static final String HINT_MESSAGE = "\n" +
112       "*****************************************\n" +
113       "Let's play Tic Tac Toe!\n" +
114       "When asked for a move, enter location you want.\n" +
115       "Enter the row first and then the column, both on the same line.\n" +
116       "The row and column must in the range 1 .. 3\n" +
117       "*****************************************\n" ;
118 }
```


TicTacToeGame.java
/**
version 1.0
The Driver class to run Tic-Tac-Toe game.
Refactored by Max He, March 2008
Rearranged on June 20, 2008
Refactored on Dec. 3, 2017
Refactored on April 26, 2018
*/
class TicTacToeGame {
public static void main (String[] args) {
new TicTacToe().play();
}
}

管理员: 命令提示符

```
H:\work\JavaProg\2018Spring\TicTacToe\v1.0>java TicTacToeGame

**************************************************
Let's play Tic Tac Toe!
When asked for a move, enter location you want.
Enter the row first and then the column, both on the same line.
The row and column must in the range 1 .. 3
**************************************************

Here is the current board:

 | |
-+-+-
 | |
-+-+-
 | |

It is your move.  You are 'X'
Enter a row and column on one line: 2 2

The move for X is 2, 2
Here is the current board:

 | |
-+-+-
 |X|
-+-+-
 | |

It is my move.  I am 'O'

The move for O is 2, 1
Here is the current board:

 | |
-+-+-
O|X|
-+-+-
 | |
```

管理员: 命令提示符

```
It is your move.  You are 'X'
Enter a row and column on one line: 1 3

The move for X is 1, 3
Here is the current board:

 | |X
-+-+-
O|X|
-+-+-
 | |

It is my move.  I am 'O'

The move for O is 3, 3
Here is the current board:

 | |X
-+-+-
O|X|
-+-+-
 | |O

It is your move.  You are 'X'
Enter a row and column on one line: 3 1

The move for X is 3, 1
Here is the current board:

 | |X
-+-+-
O|X|
-+-+-
X| |O

You won!

H:\work\JavaProg\2018Spring\TicTacToe\v1.0>
```