

The Software Crisis

Around 1960 it became clear that developing software for commerce or industry was a major problem. Software projects were usually late, over-budget and frequently unusable. This 'crisis' continues, albeit in a different form and for different reasons.

A typical survey in the late 1980s by Swanson and Lientz involved 487 software developers and found 48% of staff time was spent on maintenance, 43% on new development. At least a third of the maintenance effort was devoted to rectifying errors in the production. (Reported in D. Ince 'Software Development', 1988.) Not much change in this situation today (references).

More examples of recent problems and accidents are given in lectures.

Modern applications are likely to be very large, involving many people in their design and construction. (cf Brooks '*The Mythical Man-Month*') The focus of recent years has been on tools and development methodologies to assist the phases of analysis, design, implementation, testing and maintenance of software.

Discussion of principles and techniques of object-orientation and ADTs for the design of software that is robust, adaptable and re-usable.

The following classes are taken from David Parsons *Introductory Java* (pp84-89) with minor modifications by SBR. They illustrate how several objects can co-operate to form a program and they illustrate a number of simple design decisions. There are five classes – apart from the dice package which is imported.

```
/**
    Snake.java
    Has a constructor and access methods
*/
class Snake {

    private int head;
    private int tail;

    public Snake(int h, int t) {
        head = h;
        tail = t;
    }

    public int getHead() {
        return head;
    }
    public int getTail() {
        return tail;
    }
}

/**
    Ladder.java
*/
class Ladder {

    private int top;
    private int foot;

    public Ladder(int t, int f) {

        top = t;
        foot = f;
    }
}
```

```

    }
    public int getTop() {

        return top;
    }

    public int getFoot() {

        return foot;
    }

}

/**
    Square.java
    Represents three types of square: (i)with a
    snake's head, (ii)with ladder's foot,
    (iii) with neither.
*/

class Square {

    private boolean has_snake = false;
    private boolean has_ladder = false;

    private Snake a_snake;
    private Ladder a_ladder;

    //we may want to add a snake head

    public void addSnake(Snake s) {

        a_snake = s;
        has_snake = true;
    }

    //or add the foot of a ladder

    public void addLadder(Ladder l) {

        a_ladder = l;
        has_ladder = true;
    }
}

```

```

//methods to find if a square has a snake or a
//ladder

public boolean hasSnake()  {

    return has_snake;
}

public boolean hasLadder()  {

    return has_ladder;
}

//methods to return snake or ladder if present

public Snake getSnake()  {

    return a_snake;
}

public Ladder getLadder()  {

    return a_ladder;
}
}

/**
    Board.java
    Sets up the squares that make up the board
    together with snakes and ladders. Moves counters
    by checking for snakes and ladders.
*/

class Board  {

//the board consists of 100 squares

    private Square[]  squares;
//the constructor creates the squares and adds
//the snakes and ladders

```

```

public Board() {

//array will be one square bigger than needed so
//we can start from array element 1, ignoring
//element 0

    squares = new Square[101];

    for (int i = 1; i < 101; i++) {

        squares[i] = new Square();

    }

//add the ladders

    squares[1].addLadder(new Ladder(38,1));
    squares[4].addLadder(new Ladder(14,4));
    squares[9].addLadder(new Ladder(31,9));
    squares[21].addLadder(new Ladder(42,21));
    squares[28].addLadder(new Ladder(84,28));
    squares[36].addLadder(new Ladder(44,36));
    squares[51].addLadder(new Ladder(67,51));
    squares[71].addLadder(new Ladder(91,71));
    squares[80].addLadder(new Ladder(100,80));

//add the snakes

    squares[16].addSnake(new Snake(16,6));
    squares[47].addSnake(new Snake(47,26));
    squares[49].addSnake(new Snake(49,11));
    squares[56].addSnake(new Snake(56,53));
    squares[62].addSnake(new Snake(62,19));
    squares[64].addSnake(new Snake(64,60));
    squares[87].addSnake(new Snake(87,24));
    squares[93].addSnake(new Snake(93,73));
    squares[95].addSnake(new Snake(95,75));
    squares[98].addSnake(new Snake(98,78));

}

```

```
//now check counter position for matching head of
snake or foot of ladder
```

```
int checkForSnakesOrLadders(int counter)    {

//if square has the foot of a ladder

    if(squares[counter].hasLadder())    {

        System.out.println("On " + counter + ", up
the ladder ☺ ");
        counter =
squares[counter].getLadder().getTop();
    }

//if square has the head of a snake

    if(squares[counter].hasSnake())    {

        System.out.println("On " + counter + ",
down the snake ☹");
        counter =
squares[counter].getSnake().getTail();
    }

    return counter;

}
}
```

```

/**
    SnakesAndLadders.java
    Simulates playing the game by moving a single
    counter around the board
*/

import dice.*;

public class SnakesAndLadders {

    private int counter = 0;

    //references to Board and Dice objects

    private Board board;
    private Dice dice;

    //the constructor creates the board and dice
    //objects

    public SnakesAndLadders() {

        board = new Board();
        dice = new Dice();

    }

    //this method is a controller for playing the
    //game

    public void playGame() {

        //iterate until we reach the end (square 100)

        while (counter != 100) {

            dice.throwDice();

            //see what square this takes us to

```

```

        int nextSquare = counter +
dice.getDiceValue();

//only use the throw if it does not take us
//beyond 100 i.e. we need an exact number to
//win.

        if (nextSquare <= 100)
            counter = nextSquare;

//check to go up ladder or down snake

        if(counter < 100)  {

            counter =
board.checkForSnakesOrLadders(counter);
            System.out.println("On square " +
counter);
        }

    }

        System.out.println("counter finished on "
+ counter);
    }

//'main' creates a SnakesandLadders object and
//starts the game

    public static void main(String[] args)  {

        SnakesAndLadders myGame = new
SnakesAndLadders();
        myGame.playGame();

    }

}

```