

Learning Object-Oriented Programming, Design and TDD with Pharos

Stéphane Ducasse

March 11, 2019

Copyright 2017 by Stéphane Ducasse.

The contents of this book are protected under the Creative Commons Attribution-ShareAlike 3.0 Unported license.

You are **free**:

- to **Share**: to copy, distribute and transmit the work,
- to **Remix**: to adapt the work,

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page:

<http://creativecommons.org/licenses/by-sa/3.0/>

Any of the above conditions can be waived if you get permission from the copyright holder. Nothing in this license impairs or restricts the author's moral rights.



Your fair dealing and other rights are in no way affected by the above. This is a human-readable summary of the Legal Code (the full license):

<http://creativecommons.org/licenses/by-sa/3.0/legalcode>

Contents

Illustrations	ii
1 Snake and ladder solutions	1
1.1 A first real test	1
1.2 Accessing on tile	1
1.3 Adding players	1
1.4 Displaying players	2
1.5 Avoid leaking implementation information	2
1.6 Preparing to move players	2
1.7 Finding the tile of a player	2
1.8 Moving a player	3
1.9 New printing hook	3
1.10 Snake and ladder declaration	3
1.11 Active tile actions	4
1.12 Player turns and current player	4
1.13 Game end	4
1.14 Playing one move	4
Bibliography	5

Illustrations



Snake and ladder solutions

1.1 A first real test

```
[ SLGame >> tileNumber: aNumber  
  tiles := Array new: aNumber.  
  1 to: aNumber do: [ :i |  
    tiles at: i put: (SLTile new position: i) ]  
[ SLGame >> tileNumber  
  ^ tiles size
```

1.2 Accessing on tile

```
[ SLGame >> tileAt: aNumber  
  ^ tiles at: aNumber
```

1.3 Adding players

```
[ SLPlayer >> name: aString  
  name := aString  
[ Object subclass: #SLTile  
  instanceVariableNames: 'position players'  
  classVariableNames: ''  
  package: 'SnakesAndLadders'
```

```
[ SLGame >> initialize
  players := OrderedCollection new.

[ SLTile >> addPlayer: aPlayer
  players add: aPlayer

[ SLTile >> players
  ^ players
```

1.4 Displaying players

```
[ SLPlayer >> printOn: aStream
  aStream << '<' << name << '>'
```

1.5 Avoid leaking implementation information

```
[ SLTile >> includesPlayer: aPlayer
  ^ players includes: aPlayer
```

1.6 Preparing to move players

```
[ SLPlayer >> position
  ^ position

[ SLPlayer >> position: anInteger
  position := anInteger

[ Object subclass: #SLPlayer
  instanceVariableNames: 'name position'
  classVariableNames: ''
  package: 'SnakesAndLadders'

[ SLGame >> addPlayer: aPlayer
  aPlayer position: 1.
  (tiles at: 1) addPlayer: aPlayer

[ SLGame >> tileFor: aPlayer atDistance: aNumber
  ^ self tileAt: (aPlayer position + aNumber)
```

1.7 Finding the tile of a player

```
[ SLGame >> tileOfPlayer: aSLPlayer
  ^ tiles at: aSLPlayer position
```

1.8 Moving a player

```

[ SLTile >> removePlayer: aPlayer
  players remove: aPlayer

[ SLGame >> movePlayer: aPlayer distance: anInteger
  | targetTile |
  targetTile := self tileFor: aPlayer atDistance: anInteger.
  (self tileOfPlayer: aPlayer) removePlayer: aPlayer.
  targetTile addPlayer: aPlayer.
  aPlayer position: targetTile position.

```

1.9 New printing hook

```

[ SLAbstractTile >> printOn: aStream

  aStream << '['.
  self printInsideOn: aStream.
  aStream << ']'

[ SLLadderTile >> printInsideOn: aStream

  super printInsideOn: aStream.
  aStream << '->'.
  targetTile position printOn: aStream

```

1.10 Snake and ladder declaration

```

[ SLGame >> setSnakeFrom: aSourcePosition to: aTargetPosition

  tiles
    at: aSourcePosition
    put: (SLSnakeTile new
      position: aSourcePosition;
      to: (tiles at: aTargetPosition) ; yourself)

[ SLGame >> setLadderFrom: aSourcePosition to: aTargetPosition

  tiles
    at: aSourcePosition
    put: (SLLadderTile new
      position: aSourcePosition ;
      to: (tiles at: aTargetPosition) ; yourself)

```

1.11 Active tile actions

```
[ SLActiveTile >> acceptPlayer: aPlayer
  targetTile acceptPlayer: aPlayer
```

1.12 Player turns and current player

```
[ Object subclass: #SLGame
  instanceVariableNames: 'tiles players turn'
  classVariableNames: ''
  package: 'SnakesAndLadders'

  SLGame >> initialize
    players := OrderedCollection new.
    turn := 0
```

1.13 Game end

```
[ SLGame >> isOver
  ^ players anySatisfy: [ :each | each position = tiles size ]
```

1.14 Playing one move

```
[ SLGame >> canMoveToPosition: aNumber
  "We can only move if we stay within the game.
  This implies that the player should draw an exact number to land
  on the finish tile."

  ^ aNumber <= tiles size
```


Bibliography