P2: Exercise 2 Discussion

Yannik Dällenbach
8 March 2019
Exercise 2

SkipPlayer: Skip the next player’s turn?
  • Main problem: Find the next player

Approaches
  • Use boolean flag
  • Use an ArrayList to get random index
  • (...more advanced / dynamic solutions possible)
ScrambleUp

ScrambleUp: How to get a random player?

- Main problem: Find a random player and exchange.
public ISquare landHereOrGoHome() {
    if (this.isOccupied()) {
        return game.firstSquare();
    }
    randomPlayer = game.getRandomPlayer();
    // …
    int position = randomPlayer.position();
    // …

    return game.getSquare(position);
}
/**
 * Square that swaps player that landed on it with another random player.
 */

public class Scramble implements ISquare {
    // ...
}
The class ScrambleUp contains methods for swapping a player landing on it with another random player.
Swaps entering player with another random selected player.

Is created and called inside the `{@link Game}` class.
Extends `{@link Square}`.

If there is only one player left in the game, the player will not swapped.
These are not good commit messages:

- No more errors!
- I hate git
- FIRST TRY
- V3
- sloooowly getting there
- …
These are **better**: 

Implemented **SwapSquare**

Implemented **landHereOrGoHome** to exchange the current with another random player. **Overrides toString method**
Design by Contract, Assertions, and Exceptions
Exception or Assertion?

```java
/**
 * Sets the refresh rate for the current display.
 * @param rate new refresh rate
 */
public void setRefreshRate(int rate) {
    // what if rate < 0?
}
```
Exception or Assertion?

```java
/**
 * Sets the refresh rate for the current display.
 * @param rate new refresh rate, must be >= 0
 */
public void setRefreshRate(int rate) {
    assert rate >= 0;
}
```
/**
 * Sets the refresh rate for the current display.
 * @param rate new refresh rate
 * @throws IllegalArgumentException if rate is not valid
 */

public void setRefreshRate(int rate)
    throws IllegalArgumentException {
    if (rate < 0) {
        throw new IllegalArgumentException();
    }
}
Assertions

• Use when you expect a property to hold

• Use for contracts
  • Pre-/postconditions, invariants

• Use inside complex code
  • For example in an algorithm to make sure an intermediate result holds
Assertions

/**
 * Draw a vertical line, starting from position, with a length of steps + 1.
 * @param position start location of the line, must not be null
 * @param steps length of the line
 */
public void drawVertical(Point position, int steps) {
    assert position != null;
    // Implementation omitted
    assert (invariant());
}
Assertions

- Favor assertions/preconditions for checking method parameters in private/internal API
  - Senders come from within your project => go fix the bug!
  - Simplifies design

- Use assertions for postconditions and invariants
Exceptions

- Error handling
- Expected behavior
  - Deal with it in try-catch blocks, or
  - throw it up to the caller

```java
public void matches(String filename)
    throws NotImplementedException {

    throw new NotImplementedException();
}
```
Exceptions

Do not abuse exceptions

```java
try {
    int index = 0;
    while (true) {
        players[index++] = new Player();
    }
} catch (ArrayIndexOutOfBoundsException e) {}
```
Exceptions

Do not abuse exceptions

```java
for (int i = 0; i < players.length; i++) {
    players[i] = new Player();
}
```
Exceptions

- Favor exceptions for checking method parameters in public/external API
  - Can’t trust user to read JavaDoc

- Always use exceptions to check user input!
Checked and Unchecked Exceptions

- Checked exceptions must either be declared

```java
public void fooBar() throws TodoException { /* ... */}
```

- or wrapped inside a try-catch block

```java
public void fooBar() {
    try {
        // something that throws a TodoException
    } catch (TodoException e) {
        // handle exception
    }
}
```

- Use checked exceptions unless you have a very good reason not to!
NullPointerException

• Very common unchecked exception

• Often hard to tell where it came from
  • Value may be passed around for a while before it is used

=> Include null checks where appropriate
private void newGame() {
    setPlayer(null);
    execute();
}

private void setPlayer(Player player) {
    this.player = player;
}

private void execute() {
    this.player.move();
}
NullPointerException

Exception in thread "main" java.lang.NullPointerException at exercise_03.SomeClass.execute(SomeClass.java:79) at exercise_03.SomeClass.newGame(SomeClass.java:65) at exercise_03.SomeClass.main(SomeClass.java:7) ...
Process finished with exit code 1

Why is player == null here?
NullPointerException

private void newGame() {
    setPlayer(null);
    execute();
}

/** @param player must not be null */
private void setPlayer(Player player) {
    assert player != null;
    this.player = player;
}

private void execute() {
    this.player.move();
}

Exception in thread "main" java.lang.AssertionError
at exercise_03.SomeClass.setPlayer(SomeClass.java:74)
at exercise_03.SomeClass.newGame(SomeClass.java:64)
at exercise_03.SomeClass.main(SomeClass.java:7)
Process finished with exit code
Another example

```java
/**
 * Look up the object at the top of this stack and return it.
 * @return the object at the top
 */
public E top() {
  return top.item;
}
```

What if the stack is empty?
Another example

```java
/**
 * Look up the object at the top of
 * this stack and return it.
 * Returns null if called on an empty stack.
 * @return the object at the top
 */
public E top() {
    if (this.isEmpty()) {
        return null;
    }
    return top.item;
}
```

What if the stack contains null values?
Another example

```java
/**
* Look up the object at the top of
* this stack and return it.
* @throws EmptyStackException if the stack is empty
*
* @return the object at the top
*/
public E top() throws EmptyStackException {
    if (this.isEmpty()) {
        throw new EmptyStackException();
    }
    return top.item;
}
```
Exercise 3
Turtle Game

Demo
Turtle Game

- A turtle that moves around a 50x50 board
  - Move left, right, up, down or jump
  - Leave a red trail

- Input: String representing a turtle program

```plaintext
right 5
down 4
left 3
jump 20 20
down 10
```
Turtle Game

- You start with
  - TurtleRenderer: GUI
  - BoardMaker: Class that gets text from GUI and returns a

As always: `git pull p2-exercises master`

Read exercise_03.md

- Parse input program (split lines into commands)
- Execute turtle actions
- Keep track of trail
Main areas of application

• Documentation
• Drafts
Documentation

• Can be done automatically
• Can be an «overkill»
A draft helps you to...

... simplify reality
... understanding an existing solution
... deciding how to build something from scratch
... capture requirements and discuss your idea with others
... reduce your effort to test different approaches
Modeling your system...

structure

- class diagram
- component diagram
- composite structure diagram
- object diagram
- package diagram
- profile diagram

behaviour

- activity diagram
- communication diagram
- interaction overview diagram
- sequence diagram
- state machine diagram
- timing diagram
Modeling your system...

structure

- class diagram
- component diagram
- composite structure diagram
- object diagram
- package diagram
- profile diagram

behaviour

- activity diagram
- communication diagram
- interaction overview diagram
- sequence diagram
- state machine diagram
- timing diagram
Class diagram

Game
- squares: List(ISquare)
- players: List(Player)
- size: int
+ play(): void
+movePlayer(roll: int): void

Die uses

«interface» ISquare

Square

LastSquare

FirstSquare
Classes and Interfaces

<table>
<thead>
<tr>
<th>Name</th>
<th>Methods</th>
<th>Interface annotation</th>
</tr>
</thead>
</table>
| Game          | - squares: List(ISquare)  
                | - players: List(Player)  
                | - size: int  
                | + play(): void  
                | + movePlayer(roll: int): void  
                | «interface» ISquare |
Classes and Interfaces

<table>
<thead>
<tr>
<th>Game</th>
</tr>
</thead>
<tbody>
<tr>
<td>- squares: List(ISquare)</td>
</tr>
<tr>
<td>- players: List(Player)</td>
</tr>
<tr>
<td>- size: int</td>
</tr>
<tr>
<td>+ play(): void</td>
</tr>
<tr>
<td>+movePlayer(roll: int): void</td>
</tr>
</tbody>
</table>

Access modifiers
+ public, - private, # protected, static

Attributes
accessIdentifier: type
Example: - size: int

Methods
accessIdentifier(parameter: type): returnType
Implementation and extension

```
interface ISquare

Square

Extending a class

LastSquare

FirstSquare

Implementing an interface
```
Dependency

Game

- squares: List(ISquare)
- players: List(Player)
- size: int

+ play(): void
+ movePlayer(roll: int): void

Die

«interface» ISquare
Aggregation vs. Composition

Car

Engine

Building

Room
Keep in mind

- Different aspects, different diagram type
- Keep it simple
- Focus on what you want to communicate, forget the rest
On paper: Not enough information
On paper: Too much information

```
Game
- squares: List<ISquare>
- players: List<Player>
- size: int
- currentPlayer: Player
- winner: Player

+ isValidPosition(position: int): boolean
+ play(): void
+ notOver(): boolean
+ getSquareSize(): int
+ currentPlayer(): Player
+ movePlayer(roll: int): void
+ setSquare(position: int, square: ISquare): void
+ winner(): Player
+ toString(): String
- addSquares(size: int): void
- addPlayers(initPlayers: Player[]): void
```
On paper

Game
- squares: List(1Square)
- players: List(Player)
- size: int
+ play(die: Die)
+ movePlayer(roll: int)

<< interface >>
1Square

Die

Square
# position: int
- player: Player
+ moveAndLand(moves: int): 1Square
+ landHereOrGoHome(): 1Square
+ enter(player: Player)
+ leave(player: Player)

Player
- name: String
- square: 1Square
+ moveForward(moves: int): void

FirstSquare

LastSquare

TikTokSquare
Exercise 3

Use the information from the lecture and from this presentation to solve the UML related tasks in Exercise 3

Add both diagrams in a common format (e.g. JPG, PDF) to the exercise root in your group folder.

If you do not have a scanner, you can just take a photo of the UML diagrams with a smartphone.
To learn more

- [http://scg.unibe.ch/teaching/p2/](http://scg.unibe.ch/teaching/p2/) (P2 reading material, UML Reference)
- Book: UML Distilled, Martin Fowler