b
UNIVERSITÄT
BERN

P2: Exercise 1 Discussion

Claudio Corrodi

Two approaches

- Custom algorithm
  - Recursive
  - Look at the first character of pattern and filename at a time

- Regular expressions
  - One-liner can cover most cases
  - But: what about special characters?

Custom Algorithm using recursion

```
private boolean match(String pattern, String filename) {
    …

    // If there is another character in filename, check if it matches
    // the current pattern character. If not, the pattern does not match;
    // otherwise, check the remainder.
    if (filename.isEmpty() || pattern.charAt(0) != filename.charAt(0)) {
        return false;
    } else {
        return match(pattern.substring(1), filename.substring(1));
    }
}
```

match("abc", "abcde.txt") ==
match("bc", "bcde.txt") ==
match("c", "cde.txt") ==
match("", "de.txt") == …

Custom Algorithm using recursion

```
private boolean match(String pattern, String filename) {
    …

    // Question mark. If filename is not empty, match the remainder
    // of pattern to the remainder of filename.
    if (pattern.startsWith("?")) {
        if (filename.isEmpty()) {
            return false;
        } else {
            return match(pattern.substring(1), filename.substring(1));
        }
    }

    …
}
```

match(“?oo.txt”, “foo.txt”) == match(“oo.txt”, “oo.txt”)

Regular Expressions

```
private boolean matchRegex(String filename) {
   String regexPattern = pattern;
   regexPattern = regexPattern.replace("*", ".*");
   regexPattern = regexPattern.replace("?", ".");
   return Pattern.matches(regexPattern, filename);
}
```

"." matches exactly one character
".*" matches any number of characters

Regular Expressions

```java
private boolean matchRegex(String filename) {
    String regexPattern = pattern;
    regexPattern = regexPattern.replace("*", ".*");
    regexPattern = regexPattern.replace("?", ".");
    return Pattern.matches(regexPattern, filename);
}
```

"." matches exactly one character
".*" matches any number of characters

- What about special characters?

  ➔ Read documentation!

```java
    // escape special character "."
    regexPattern = regexPattern.replace(".", "\\.");
```

Examples: Encapsulation & names

```
public class FilePattern implements FileFilter {

    public String string;

    public FilePattern(String string) {
        this.string = string;
    }

    ...
}
```

Examples: Encapsulation & names

```
public class FilePattern implements FileFilter {

    public String string;

    public FilePattern(String string) {
        this.string = string;
    }

    ...
}
```

Examples: Encapsulation & names

```
public class FilePattern implements FileFilter {

    private String pattern;

    public FilePattern(String pattern) {
        this.pattern = pattern;
    }

    ...
}
```

Make attributes private

Use meaningful names

Examples: Useless code

```java
private String tempPattern;

public String getTempPattern() {
   return this.tempPattern;
}
```

Examples: Useless code

```
private String tempPattern;

~~public String getTempPattern() {~~
~~    return this.tempPattern;~~
~~}~~
```

Unused outside of class!

Manual Testing

```
public class TestMain {
   public static void main(String[] args) {
      FilePattern a = new FilePattern("fname*");
      System.out.println(a.accept(new File("")));
   }
}
```

Manual Testing

```
public class TestMain {
   public static void main(String[] args) {
      FilePattern a = new FilePattern("fname*");
      System.out.println(a.accept(new File("")));
   }
}
```

```
public class FilePatternTest {
   …

   @Test
   public void fnameStarDoesNotMatchEmptyName() {
      FilePattern a = new FilePattern("fname*");
      assertFalse(a.accept(new File("")));
   }
}
```

add scenario as a **permanent** test

UNIVERSITÄT BERN

P2: Exercise 2

Claudio Corrodi

Exercise 2: Snakes & Ladders

- You are given a skeleton for the Snakes & Ladders game

- Add new types of squares
  - TikTokSquare: “Ladder” with two alternating destinations
  - SwapSquare: When landing here, swap your position with another player
  - …

- *Test behavior of squares*
  - *Use JUnit and JExample*

- *Write proper documentation*

JUnit, JExample

- Testing frameworks
  - Covered in more detail in lecture 4!
- Goal: Make sure program behaves as expected
- **JUnit**: Individual, independent tests
- **JExample**: Sequences of tests
  - Maintain state between tests
  - No need to reinitialize

JUnit

```java
@Test
public void newGame() {
   jack = new Player("Jack");
   jill = new Player("Jill");
   Player[] args = { jack, jill };
   Game game = new Game(12, args);
   game.setSquareToLadder(2, 4);
   game.setSquareToLadder(7, 2);
   game.setSquareToSnake(11, -6);

   assertTrue(game.notOver());
   assertEquals(1, jack.position());
   assertEquals(1, jill.position());
   assertEquals(jack, game.currentPlayer());
}
```

```java
@Test
public void initialStrings() {
   jack = new Player("Jack");
   jill = new Player("Jill");
   Player[] args = { jack, jill };
   Game game = new Game(12, args);
   game.setSquareToLadder(2, 4);
   game.setSquareToLadder(7, 2);
   game.setSquareToSnake(11, -6);

   assertEquals("Jack", jack.toString());
   assertEquals("Jill", jill.toString());
   assertEquals("[1<Jack><Jill>]",
        game.firstSquare().toString());
}
```

JUnit

```java
@Test
public void newGame() {
   jack = new Player("Jack");
   jill = new Player("Jill");
   Player[] args = { jack, jill };
   Game game = new Game(12, args);
   game.setSquareToLadder(2, 4);
   game.setSquareToLadder(7, 2);
   game.setSquareToSnake(11, -6);

   assertTrue(game.notOver());
   assertEquals(1, jack.position());
   assertEquals(1, jill.position());
   assertEquals(jack, game.currentPlayer());
}
```

```java
@Test
public void initialStrings() {
   jack = new Player("Jack");
   jill = new Player("Jill");
   Player[] args = { jack, jill };
   Game game = new Game(12, args);
   game.setSquareToLadder(2, 4);
   game.setSquareToLadder(7, 2);
   game.setSquareToSnake(11, -6);

   assertEquals("Jack", jack.toString());
   assertEquals("Jill", jill.toString());
   assertEquals("[1<Jack><Jill>]",
         game.firstSquare().toString());
}
```

JExample

```java
@Test
public Game newGame() {
   jack = new Player("Jack");
   jill = new Player("Jill");
   Player[] args = { jack, jill };
   Game game = new Game(12, args);
   game.setSquareToLadder(2, 4);
   assertTrue(game.firstSquare().isOccupied());
   return game;
}
```

JExample

```
@Test
public Game newGame() {
   jack = new Player("Jack");
   jill = new Player("Jill");
   Player[] args = { jack, jill };
   Game game = new Game(12, args);
   game.setSquareToLadder(2, 4);
   assertTrue(game.firstSquare().isOccupied());
   return game;
}
```

```
@Given("newGame")
public Game initialPositions(Game game) {
   assertEquals(1, jack.position());
   assertEquals(1, jill.position());
   return game;
}
```

JExample

```
@Test
public Game newGame() {
   jack = new Player("Jack");
   jill = new Player("Jill");
   Player[] args = { jack, jill };
   Game game = new Game(12, args);
   game.setSquareToLadder(2, 4);
   assertTrue(game.firstSquare().isOccupied());
   return game;
}
```

```
@Given("newGame")
public Game initialPositions(Game game) {
   assertEquals(1, jack.position());
   assertEquals(1, jill.position());
   return game;
}
```

```
@Given(“initialPositions")
public Game move1jack(Game game) {
   game.movePlayer(4);
   assertTrue(game.notOver());
   assertEquals(5, jack.position());
   assertEquals(1, jill.position());
   assertEquals(jill, game.currentPlayer());
   return game;
}
```

JExample

```java
@Test
public Game newGame() {
   jack = new Player("Jack");
   jill = new Player("Jill");
   Player[] args = { jack, jill };
   Game game = new Game(12, args);
   game.setSquareToLadder(2, 4);
   assertTrue(game.firstSquare().isOccupied());
   return game;
```

```java
@Given("newGame")
public Game initia
   assertEquals(1,
   assertEquals(1,
   return game;
}
```

More in exercise_02.md

git pull p2ubungen master

```java
@Given("initialPositions")
public Game move1jack(Game game) {
   game.movePlayer(4);
   assertTrue(game.notOver());
   assertEquals(5, jack.position());
   assertEquals(1, jill.position());
   assertEquals(jill, game.currentPlayer());
   return game;
}
```

License

- http://creativecommons.org/licenses/by-sa/2.5/