軟體測試 lab 4

網工一

309552026

鄭偉丞

Github url: https://github.com/jordan0210/st_nycu_lab4_309552026

Description:

Part 1:

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args){
        Game game = new Game();

        Scanner input = new Scanner(System.in);
        System.out.println("Enter Player 1 choice (rock, paper or scissors):");
        String player1 = input.next();
        game.player[0] = game.validation(player1);

        System.out.println("Enter Player 2 choice (rock, paper or scissors):");
        String player2 = input.next();
        game.player[1] = game.validation(player2);

        game.checkWinner();
    }
}
```

```java
public class Game {
    private int[][] winnerArray = {{0, 2, 1},
                                   {1, 0, 2},
                                   {2, 1, 0}};
    int[] player = new int[2];

    public Game() { System.out.println("=== Welcome to Rock-Paper-Scissors game ==="); }

    public void checkWinner(){
        switch(winnerArray[player[0]][player[1]]){
            case 0:
                System.out.println("Draw!");
                break;
            case 1:
                System.out.println("Player 1 win!");
                break;
            case 2:
                System.out.println("Player 2 win!");
                break;
        }
    }

    public int validation(String input) throws IllegalArgumentException{
        switch(input){
            case "rock":
                return 0;
            case "paper":
                return 1;
            case "scissors":
                return 2;
            default:
                throw new IllegalArgumentException("Bad Choice!");
        }
    }
}
```

main 主要控制遊戲的流程讀取 User 的輸入,Game 負責判斷 User 的輸入是否合法,不合法的輸入會觸發 exception,最後用 winnerArray 來判斷誰贏.

Part2:

```java
private static Stream<Arguments> AllWinTestInputParameter(){
    return Stream.of(
            Arguments.of(new int[]{0, 2}, "Player 1 win!\n"),
            Arguments.of(new int[]{1, 0}, "Player 1 win!\n"),
            Arguments.of(new int[]{2, 1}, "Player 1 win!\n")
    );
}
private static Stream<Arguments> AllLoseTestInputParameter(){
    return Stream.of(
            Arguments.of(new int[]{0, 1}, "Player 2 win!\n"),
            Arguments.of(new int[]{1, 2}, "Player 2 win!\n"),
            Arguments.of(new int[]{2, 0}, "Player 2 win!\n")
    );
}
private static Stream<Arguments> AllDrawTestInputParameter(){
    return Stream.of(
            Arguments.of(new int[]{0, 0}, "Draw!\n"),
            Arguments.of(new int[]{1, 1}, "Draw!\n"),
            Arguments.of(new int[]{2, 2}, "Draw!\n")
    );
}
private static Stream<Arguments> TestValidInputParameter(){
    return Stream.of(
            Arguments.of("rock", 0),
            Arguments.of("paper", 1),
            Arguments.of("scissors", 2)
    );
}
private static Stream<Arguments> TestInvalidInputParameter(){
    return Stream.of(
            Arguments.of("123123"),
            Arguments.of("Rock"),
            Arguments.of("ppaper"),
            Arguments.of("Scisors")
    );
}
```

圖中是 test case,分別為 Player 1 win, Player 2 win, Draw, 合法/不合法輸入,每個測

資都有相對應的 testfunction.

Part 3, Part 4:
github workflow url:
https://github.com/jordan0210/st_nycu_lab4_309552026/blob/main/.github/workflows/maven.yml

```yaml
Compile:
  runs-on: ubuntu-latest

  steps:
  - uses: actions/checkout@v2
  - name: Set up JDK 1.8
    uses: actions/setup-java@v1
    with:
      java-version: 1.8
  - name: Compile with Maven
    run: mvn clean -B compile
```

用設定 java version 並用 mvn compile 做 Compile,這個動作不會做 test

```yaml
Test:
  needs: Compile
  runs-on: ubuntu-latest

  steps:
  - uses: actions/checkout@v2
  - name: Set up JDK 1.8
    uses: actions/setup-java@v1
    with:
      java-version: 1.8
  - name: Test with Maven
    run: mvn -B test --log-file testReport.txt
  - name: Update Test Log
    uses: actions/upload-artifact@v2
    with:
      name: testReport
      path: testReport.txt
```

設定環境,用 mvn test 做 test, 並將 log 輸出成 testReport.txt, 再透過 upload-artifact 上傳 report

```yaml
steps:
  - uses: actions/checkout@v2
  - name: Set up JDK 1.8
    uses: actions/setup-java@v1
    with:
      java-version: 1.8
  - name: Get the version
    id: get_version
    run: echo ::set-output name=VERSION::$(echo $GITHUB_REF | cut -d / -f 3)
  - name: Update Release Version in pom.xml
    run: mvn -B versions:set -DnewVersion=${{ steps.get_version.outputs.VERSION }}-SNAPSHOT -DgenerateBackupPoms=false
  - name: Assemble Jar File
    run: mvn -B clean compile assembly:single
  - name: Update Assemble Jar File
    uses: actions/upload-artifact@v2
    with:
      name: buildJar
      path: ./target/st_nycu_lab4_309552026-${{ steps.get_version.outputs.VERSION }}-SNAPSHOT-jar-with-dependencies.jar
```

先設定環境, Get version 這一步取得這 commit 的 tag(如果沒有 tag 會用 branch 的名字), 然後透過 mvn version:set 更新 pom.xml 中的 version, 用 mvn compile assembly:single 依據 pom.xml 做 assemble, 因為上一步修改過 version,所以這邊的 version 會是 push 上去的 tag 的 version, 最後用 upload-artifact 上傳 jar file.

```yaml
Release:
  if: startsWith(github.ref, 'refs/tags/')
  needs: Assemble
  runs-on: ubuntu-latest

  steps:
    - uses: actions/checkout@v2
    - name: Download Artifact
      uses: actions/download-artifact@v2
      with:
        name: buildJar
    - name: Release
      uses: softprops/action-gh-release@v1
      with:
        files: ./*.jar
      env:
        GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

先透過 if,判斷是否有 tag,如果沒有 tag 則會直接填過這個 job.確認完之後先 download Assemble 所上傳的 jar file, 再透過 action-gh-release 做 release.