SE-Project - 1

Bonus Task: Automated Refactoring Pipeline

Team - 22

Gunank Singh Jakhar (2022201057)

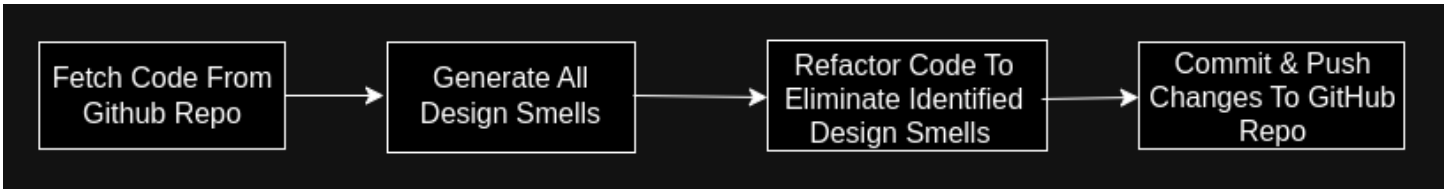
M Elamparithy (2022801014)

Rishabh Gupta (2022202011)

Piyush Singh (2022201032)

Prashant Kumar (2022201058)

Flow Chart:



Task-1: (Automated Design Smell Detection)

Libraries used

```
import openai
import time
import os
from github import Github, RateLimitExceededException
import requests
import base64
import re
import subprocess
```

Fetch code from GitHub

```python
def get_code_from_github(repo_url, access_token):
    try:
        gh = Github(access_token)
        repo = gh.get_repo(repo_url)
        contents = repo.get_contents("")  # Get root directory contents
        code = []
        for content in contents:
            if content.type == "file":
                file_content = requests.get(content.download_url).text
                code.append(file_content.decode("utf-8"))
        return "\n".join(code)
    except RateLimitExceededException as e:
        print(f"GitHub API rate limit exceeded: {e}")
        time.sleep(60)  # Wait for a minute before retrying
        return get_code_from_github(repo_url, access_token)
    except Exception as e:
        print(f"Error fetching code from GitHub: {e}")
        return None
```

Fetch all Java files from the GitHub Repository

```python
def fetch_all_java_files(repo_owner, repo_name, branch="main"):
    # GitHub API endpoint for getting contents of a directory
    url = f"https://api.github.com/repos/{repo_owner}/{repo_name}/contents?ref={branch}"
    headers = {
        "Authorization": f"Bearer {GITHUB_TOKEN}",
    }
    # Make a GET request to the GitHub API
    response = requests.get(url, headers=headers)

    if response.status_code == 200:
        # Parse the JSON response
        contents = response.json()

        # Iterate through each item in the directory
        # java_files_content = {}
        code = []
        for content_info in contents:
            if content_info['type'] == 'file' and content_info['name'].endswith('.java'):
                # Fetch content for each Java file
                file_path = content_info['path']
                file_content = fetch_file_content(repo_owner, repo_name, file_path, branch)
                # java_files_content[file_path] = file_content
                code.append(file_content)

            return "\n".join(code)
    else:
        print(f"Error: Unable to fetch file list. Status code: {response.status_code}")
        return None
```

Extract code from Java file

```python
def fetch_file_content(repo_owner, repo_name, file_path, branch="main"):
    # GitHub API endpoint for getting content
    url = f"https://api.github.com/repos/{repo_owner}/{repo_name}/contents/{file_path}?ref={branch}"
    headers = {
        "Authorization": f"Bearer {GITHUB_TOKEN}",
    }
    # Make a GET request to the GitHub API
    response = requests.get(url, headers=headers)

    if response.status_code == 200:
        # Parse the JSON response
        content_info = response.json()

        # # Decode the content from base64
        file_content = content_info.get('content', '')
        file_content = file_content.encode('utf-8')
        file_content = base64.b64decode(file_content).decode('utf-8')

        return file_content
    else:
        print(f"Error: Unable to fetch file content. Status code: {response.status_code}")
        return None
```

Ask chatGPT to identify the design smells in the code using appropriate prompt message and openAI api

```python
def generate_description(code):
    # Use OpenAI API to generate design smells in code
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages= [{"role": "system", "content": f"Detect potential design smells in the following java code:\n\n{code}\n"}],
        max_tokens=500
    )
    res_msg = response.choices[0].message['content'].strip()
    return re.findall(r"\*\*(.*?)\*\*", res_msg)
```

Task-2: (Automated Refactoring)

Ask chatGPT to generate refactored code taking the identified design smells into account (again using appropriate prompt message which includes the code snippet and the identified design smells)

```python
def refactor_code(design_smells_list, code):
    design_smells = ",".join(design_smells_list)
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages= [{"role": "system", "content": f"Refactor the below java code to eliminate the following design smells '{design_smells}' \n\n{c
        max_tokens=500
    )
    return response.choices[0].message['content'].strip().strip("```")[4:]
```

Clone/ update the local repository

```python
def clone_or_update_repo(repo_url, local_folder):
    repo_path = local_folder+"\SE_test"
    if os.path.exists(repo_path):
        print(f"Repository already exists in {local_folder}. Updating...")
        update_repo(repo_path)
    else:
        clone_repo(repo_url, local_folder)

def clone_repo(repo_url, local_folder):
    try:
        subprocess.run(['git', 'clone', repo_url, local_folder], check=True)
        print(f"Repository cloned successfully to {local_folder}")
    except subprocess.CalledProcessError as e:
        print(f"Error: {e}")

def update_repo(local_folder):
    try:
        subprocess.run(['git', 'pull'], cwd=local_folder, check=True)
        print(f"Repository updated successfully in {local_folder}")
    except subprocess.CalledProcessError as e:
        print(f"Error: {e}")
```

Overwrite the refactored code in the local repository

```python
def overwrite_refactored_code(java_file_path, refactored_code):
    with open(java_file_path, "w") as file:
        file.write(refactored_code)
```

Task-3: (Pull Request Generation)

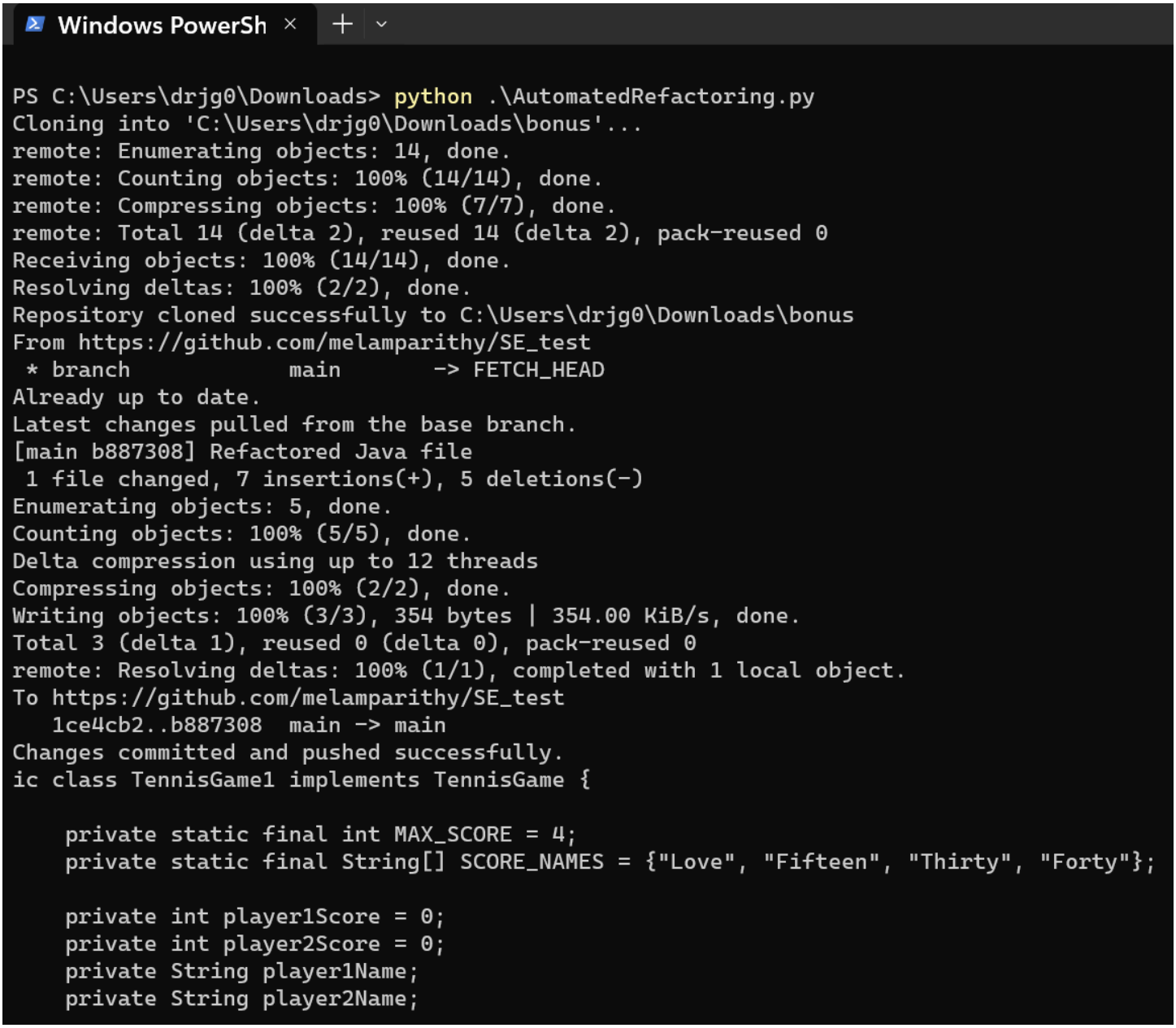
Take latest pull from GitHub and then push updated refactored code to GitHub

```python
def push_changes(java_file_path, commit_message="Refactored Java file"):
    try:
        # Fetch latest code
        subprocess.run(['git', 'pull', 'origin', 'main'])
        print("Latest changes pulled from the base branch.")

        # Add the modified file to the staging area
        subprocess.run(['git', 'add', java_file_path], check=True)

        # Commit the changes
        subprocess.run(['git', 'commit', '-m', commit_message], check=True)

        # Push the changes to the remote repository
        subprocess.run(['git', 'push', 'origin', 'main'], check=True)

        print("Changes committed and pushed successfully.")
    except subprocess.CalledProcessError as e:
        print(f"Error: {e}")
```

Main function (demonstrates the complete pipeline)

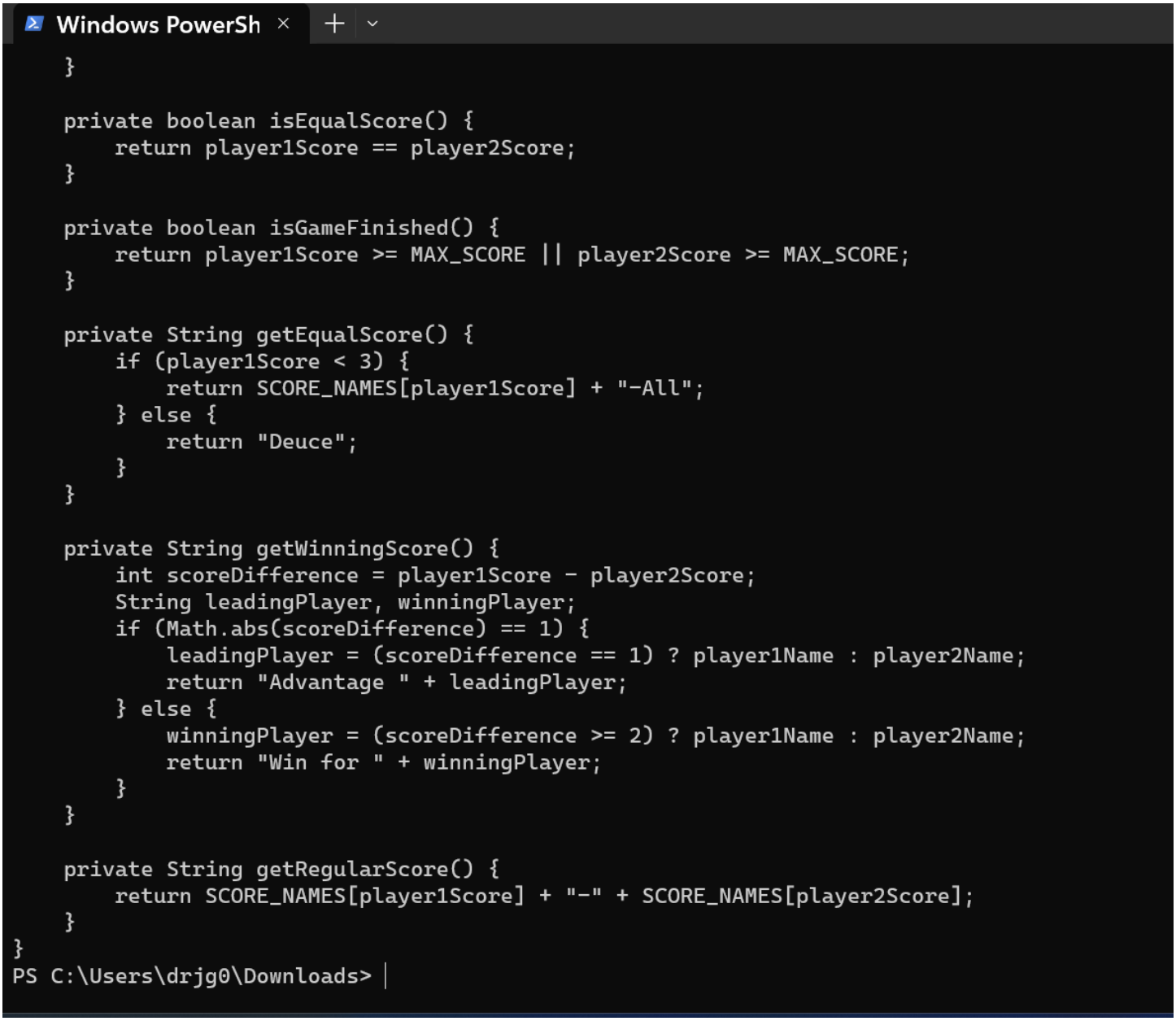
```python
if __name__ == "__main__":
    # Replace these values with the GitHub repository URL and local folder path
    local_repo_path = r"C:\Users\drjg0\Downloads\bonus"
    clone_or_update_repo(REPO_URL, local_repo_path)
    os.chdir(local_repo_path)
    java_file_path = local_repo_path + "\TennisGame1.java"

    code = fetch_all_java_files("melamparithy", "SE_test")
    design_smells = generate_description(code)
    refactored_code = refactor_code(design_smells, code)

    overwrite_refactored_code(java_file_path, refactored_code)
    push_changes(java_file_path)

    print(refactored_code)
```

Working Demo (Refactored Code in Command Prompt):


Windows PowerSh
PS C:\Users\drjg0\Downloads> python .\AutomatedRefactoring.py
Cloning into 'C:\Users\drjg0\Downloads\bonus'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 14 (delta 2), reused 14 (delta 2), pack-reused 0
Receiving objects: 100% (14/14), done.
Resolving deltas: 100% (2/2), done.
Repository cloned successfully to C:\Users\drjg0\Downloads\bonus
From https://github.com/melamparithy/SE_test
* branch main -> FETCH_HEAD
Already up to date.
Latest changes pulled from the base branch.
[main b887308] Refactored Java file
1 file changed, 7 insertions(+), 5 deletions(-)
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 354 bytes | 354.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/melamparithy/SE_test
1ce4cb2..b887308 main -> main
Changes committed and pushed successfully.
ic class TennisGame1 implements TennisGame {
private static final int MAX_SCORE = 4;
private static final String[] SCORE_NAMES = {"Love", "Fifteen", "Thirty", "Forty"};
private int player1Score = 0;
private int player2Score = 0;
private String player1Name;
private String player2Name;

Windows PowerSh

```
    private String player1Name;
    private String player2Name;

    public TennisGame1(String player1Name, String player2Name) {
        this.player1Name = player1Name;
        this.player2Name = player2Name;
    }

    @Override
    public void wonPoint(String playerName) {
        if (isPlayer1(playerName)) {
            player1Score++;
        } else {
            player2Score++;
        }
    }

    private boolean isPlayer1(String playerName) {
        return playerName.equals(player1Name);
    }

    @Override
    public String getScore() {
        if (isEqualScore()) {
            return getEqualScore();
        } else if (isGameFinished()) {
            return getWinningScore();
        } else {
            return getRegularScore();
        }
    }

    private boolean isEqualScore() {
        return player1Score == player2Score;
    }
```



Assumption: For simplicity, we demonstrated our refactoring pipeline on a sample Java file - TennisGame1.java file in order not to disturb the working code in the project-1 repo, since ChatGPT, though it gives compilation error-free and functionality-preserving code it's not always working necessarily.

Link: https://github.com/melamparithy/SE_test