Exercise

Exercise 1

In this GitHub Action exercise will utilize both self-hosted and GitHub-hosted runners for versatile workflow execution. Leverage marketplace actions to configure Java environments seamlessly. Filter Activity Type for pull_request events, focusing on both opening and closing actions for streamlined workflow management. This comprehensive approach equips candidates with the skills to effectively deploy workflows tailored to specific event triggers in GitHub repositories.

```yaml
name: Java CI with Pull Requests

on:
  pull_request:
    types: [opened, closed]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Set up JDK
      uses: actions/setup-java@v2
      with:
        java-version: '11'

    - name: Build with Maven
      run: mvn -B package --file pom.xml

  test:
    runs-on: self-hosted

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Set up JDK
      uses: actions/setup-java@v2
      with:
        java-version: '11'

    - name: Test with Maven
      run: mvn -B test
```

Exercise 2

In this GitHub Action exercise, students will leverage GitHub Context to access environment variables and secrets using the env context, ensuring secure and dynamic workflow configuration. They'll employ Expressions for dynamic variable assignment and conditional steps, enabling adaptive workflow behavior based on contextual data. By utilizing organization, repository, and environment variables, students will create versatile workflows tailored to specific project requirements. Furthermore, they'll utilize status check functions to perform essential actions based on workflow execution status, facilitating efficient workflow management and automation.

```yaml
name: Java CI/CD

on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Set up JDK
      uses: actions/setup-java@v2
      with:
        java-version: ${{ env.JAVA_VERSION }}

    - name: Build with Maven
      run: mvn -B package --file pom.xml

  test:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Set up JDK
      uses: actions/setup-java@v2
      with:
        java-version: ${{ env.JAVA_VERSION }}

    - name: Test with Maven
      run: mvn -B test

  deploy:
    runs-on: self-hosted
    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Deploy to staging
      if: ${{ github.event_name == 'push' && github.ref == 'refs/heads/main' }}
      run: |
        # Add deployment steps here
        echo "Deploying to staging environment"

    - name: Notify status
      if: ${{ always() }}
      run: |
        if [ ${{ job.status }} == 'success' ]; then
          echo "All checks passed!"
        else
          echo "One or more checks failed. Please check the workflow logs for
details."
        fi
```

Exercise 3

In this GitHub Action exercise will orchestrate a three-job workflow where jobs 2 and 3 depend on the completion of the first job. They will dynamically select the Java version as an input parameter, ensuring flexibility in their workflow configuration. Outputs from the initial setup job will propagate Java version selections to subsequent jobs, facilitating seamless integration across the workflow. Through this structured approach, students will gain proficiency in managing execution flow and leveraging inputs and outputs effectively in GitHub Actions.

```
name: Java CI/CD Workflow

on:
  workflow_dispatch:
    inputs:
      java-version:
        description: 'Java version to use'
        required: true
        default: '11'
        options: ['8', '11', '15']

jobs:
  setup:
    runs-on: ubuntu-latest

    outputs:
      java-version: ${{ steps.set_java.outputs.java-version }}

    steps:
    - name: Set up Java
      id: set_java
      run: |
        echo "::set-output name=java-version::${{ github.event.inputs.java-version
}}"

  build:
    runs-on: ubuntu-latest
    needs: setup

    strategy:
      matrix:
        java-version: ${{fromJson(needs.setup.outputs.java-version)}}

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Set up JDK
      uses: actions/setup-java@v2
      with:
        java-version: ${{ matrix.java-version }}

    - name: Build with Maven
      run: mvn -B package --file pom.xml

  test:
    runs-on: ubuntu-latest
    needs: setup

    strategy:
      matrix:
        java-version: ${{fromJson(needs.setup.outputs.java-version)}}

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Set up JDK
      uses: actions/setup-java@v2
      with:
        java-version: ${{ matrix.java-version }}

    - name: Test with Maven
      run: mvn -B test
```