

Завдання 1

Дане завдання призначене для роботи з масивами, циклами та умовними конструкціями. А також для ознайомлення зі стандартним циклом розробки з використанням системи контролю версій GitHub та системи постійної збірки (Continuous Integration - CI).

Система постійної збірки (CI) призначена для аналізу якості вихідного коду, його компіляції, та запуску тестів. Код для збірки CI бере з репозиторію. Даний підхід є обов'язковий при командній роботі над проектом.

Інструкція:

1. Скачайте [архів](#) з шаблоном проекту *TemperatureSeries* та розархівуйте його
2. Додайте його в локальний *Git* репозиторій
 - a. Перейдіть в командному рядку до директорії *TemperatureSeries*
 - b. Виконайте команди:

```
> git init
> git add *
> git commit -m "TemperatureSeries initial commit"
```
3. Створіть на своєму GitHub новий репозиторій з іменем за наступним шаблоном `cs16<surname>-hw1` (наприклад [apps19dobosevych-hw1](#))
4. Знову перейдіть в директорію *TemperatureSeries* і виконайте наступну команду

```
> git remote add origin https://github.com/<user_name>/apps19<surname>-hw1.git
```

(наприклад,
`git remote add origin https://github.com/dobosevych/apps19dobosevych-hw1.git`
)
5. Виконайте команду

```
> git push -u origin master
```

Після цього може відкрити проект в IDE і почати виконувати завдання

Важливо: <https://github.com/dobosevych/apps19dobosevych-hw1.git>

- не змінюйте структуру шаблонного проекту та не видаляйте файли *checkstyle.xml*, *pom.xml*
- класи які необхідно редагувати під час виконання завдання:
 - *TemperatureSeries/src/main/java/ua/edu/ucu/tempseries/TemperatureSeriesAnalysis.java*
 - *TemperatureSeries/src/main/java/ua/edu/ucu/tempseries/TempSummaryStatistics.java*
 - *TemperatureSeries/src/test/java/ua/edu/ucu/tempseries/TemperatureSeriesAnalysisTest.java*
- при коміті проекту в локальний репозиторій (та на GitHub) необхідно щоб виключно були закомічені: *src/*, *checkstyle.xml*, *pom.xml*. Директорію *target/*, яка буде створена під час компіляції комітити не треба.

Завдання:

Реалізувати наступні методи для класу *TemperatureSeriesAnalysis* по роботі з рядом значень температур.

Написати тести у *TemperatureSeriesAnalysisTest.java* (по аналогії з тими що є у прикладі) для перевірки коректності реалізованих методів.

Опис методів наведено нижче.

- *double average()*

Обчислює середнє значення температури. Якщо ряд порожній, генерує *IllegalArgumentException*.

- *double deviation()*

Обчислює середньоквадратичне відхилення. Якщо ряд порожній, генерує *IllegalArgumentException*.

- *double min()*

Повертає мінімальну температуру. Якщо ряд порожній, генерує *IllegalArgumentException*.

- *double max()*

Повертає максимальну температуру. Якщо ряд порожній, генерує *IllegalArgumentException*.

- *double findTempClosestToZero()*

Повертає значення температури найближче до 0. Якщо ряд порожній, генерує *IllegalArgumentException*.

Якщо у ряді є декілька значень однаково наближених до 0 (наприклад -0.2 і 0.2), то повертається додатнє значення (тобто 0.2)

- *double findTempClosestToValue(double tempValue)*

Аналогічно попередньому методі, тільки повертається значення найближче до заданого *tempValue*

- *double[] findTempsLessThan(double tempValue)*

Повертає масив зі значеннями температури менше заданого *tempValue*.

- *double[] findTempsGreaterThan(double tempValue)*

Повертає масив зі значеннями температури більше або рівне заданому *tempValue*.

- *TempSummaryStatistics summaryStatistics()*

Повертає *immutable* екземпляр класу *TempSummaryStatistics* в якому міститься інформація:

- *double avgTemp*;
- *double devTemp*;
- *double minTemp*;
- *double maxTemp*;

Якщо ряд порожній, генерує *IllegalArgumentException*.

- *int addTemps(double ... temps)*

Додає в кінець ряду вже існуючих даних нові значення температур, і повертає сумарне число значень температур. Структура (масив) в класі *TemperatureSeriesAnalysis* для зберігання вже переданих температур повинна збільшуватися в 2 рази, якщо в ній немає місця для зберігання нових значень.

Додаткові вимоги:

- клас *TemperatureSeriesAnalysis* повинен мати конструктор за замовченням та конструктор з параметром який приймає початковий ряд температур
- якщо в переданому ряді температур, зустрічається хоча б одне значення менше ніж -273C, то все значення з даного ряду не повинні додаватися до основного ряду і має генеруватись *InputMismatchException* (*throw new InputMismatchException()*)
- при реалізації не використовувати динамічні масиви (*ArrayList*) і зв'язані списки (*LinkedList*)


Робота с системою CI Hudson/Jenkins

1. Перейдіть за посиланням <http://104.248.142.53:8080/>
2. Увійдіть у систему CI під логіном **"apps19<surname>"** та паролем **"ucu312"** (можете його змінити)
3. Виберіть та натисніть зліва посилання **New Item**
4. У формі що відкрилась:
 - a. у полі **Enter an item name** введіть ім'я аналогічне назві репозиторію на GitHub - **apps19<surname>-hw1**
 - b. виберіть останній пункт **Copy existing job**
 - c. введіть у полі **Copy from** - **apps19dobochevych-hw1**
 - d. Натисніть Ок

Enter an item name

» Required field

If you want to create a new item from other existing, you can use this option:

 Copy from

5. У результаті має відкритись форма з детальним налаштуванням завдання яке ви створюєте. Вам в ньому необхідно лише вказати посилання на GitHub репозиторій з якого система CI буде брати код для компіляції та перевірки.
6. У розділі **Source Code Management** змініть значення *URL of repository* на ваш репозиторій https://github.com/<user_name>/apps19<surname>-hw1.git

Управління вихідним кодом

☐ None
☒ Git

Repositories

Repository URL

Credentials

- У самому низу натисніть кнопку Save. Після цього має створитись нова Job, завданням якої буде брати з вашого репозиторію вихідний код, компілювати його, виконувати тести та перевіряти якість коду. Натисніть у вікні кнопку **Enable** для активації Job.

Maven project apps19dobosevych-hw1

[Додати опис](#)

- Натиснувши **Build Now** ви розпочнете збірку проекту

[Back to Main Dashboard](#)
[Status](#)
[Changes](#)
[Workspace](#)
[Build Now](#)
[Delete Job](#)
[Configure](#)
[Coverage Trend](#)

Build History (trend)

#1 [Oct 2, 2016 7:43:09 PM](#)

[for all](#)
[for failures](#)

- Якщо збірка пройшла без помилок ви побачите зелене коло, якщо ні - то червоне. При натисканні на напису біля нього чи на значку консолі, ви можете побачити причини невдалої збірки.
- Якщо збірка пройшла вдало, то розділі меню зліва з'являться додаткові пункти. Нас будуть цікавити *Static Analysis Warnings* та *Coverage Report*. *Static Analysis Warnings* - буде показувати проблеми з вашим кодом. *Coverage Report* - позиватиме на скільки гарно ваші тести перевіряють ваш код. В ідеалі немає бути повідомлень про проблеми з кодом, а покриття тестами має прямувати до 100%

[Back to Job Dashboard](#)

[Status](#)

[Changes](#)

[Build Now](#)

[Console Output](#)

[Configure Build](#)

[Configure Job](#)

[Git Build Data](#)

[Maven 3](#)

[FindBugs Warnings](#)

[PMD Warnings](#)

[Static Analysis Warnings](#)

[Coverage Report](#)

[Previous Build](#)

[Back to Job Dashboard](#)

[Status](#)

[Changes](#)

[Build Now](#)

[Console Output](#)

[Configure Build](#)

[Configure Job](#)

[Git Build Data](#)

[Maven 3](#)

[FindBugs Warnings](#)

[PMD Warnings](#)

[Static Analysis Warnings](#)

[Coverage Report](#)

[Previous Build](#)

Static Analysis Result

Warnings Trend

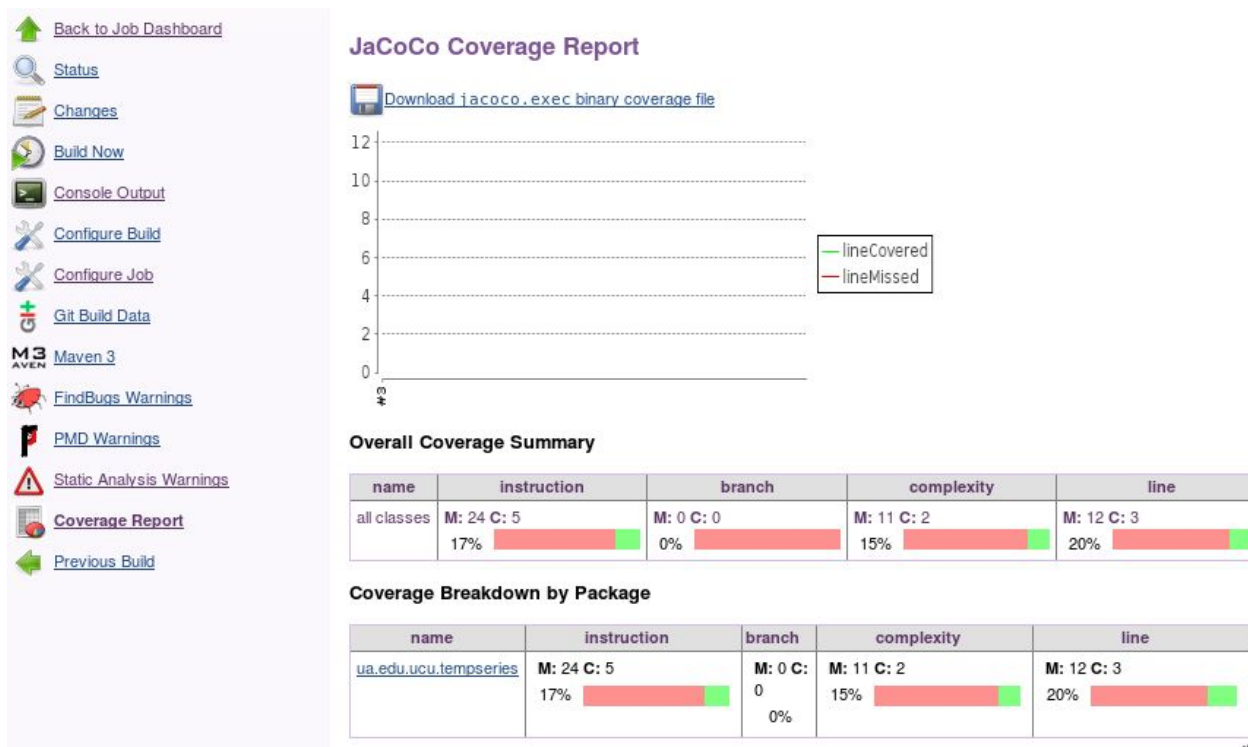
All Warnings	New Warnings	Fixed Warnings
3	<u>3</u>	0

Summary

Total	High Priority	Normal Priority	Low Priority
3	0	1	<u>2</u>

Details

Modules	Categories	Types	Warnings	Details	New	Normal	Low
Module	Total	Distribution					
Default Module	1	<div></div>					
TemperatureSeries	2	<div></div>					
Total	3						



11. Після того як ви вносите виправлення чи зміни до вашого коду, ви маєте його повторно закомітити та у системі CI знову запустити збірку проекту.