GitHub Repository: github.com/tchk7/449

Sprint #0 Report

Instructions

Objectives

- Make decisions on the SOS software development project.
- Learn unit testing and GUI programming in the language of your choice.

Deliverables and Grading Policy

Read the "CS 449 Homework Overview" document **carefully** and make the key decisions for the software development. Use the following template to complete your report.

1. Key Decisions of the SOS Project (2 points)

Object-oriented programming language	Python
GUI library (strongly encouraged)	PySide
IDE (Integrated Development Environment)	PyCharm
xUnit framework (e.g., JUnit for Java)	pytest
Programming style guide (must read it carefully)	python
Project hosting site	Github.com
Other decisions if applicable	

Sample programming style guides:

- Google Java Style Guide: https://google.github.io/styleguide/javaguide.html
- Google C++ Style Guide: https://google.github.io/styleguide/cppguide.html
- Google Python Style Guide: https://google.github.io/styleguide/pyguide.html

2. Unit testing (4 points)

Find a tutorial on the unit test framework you have chosen and write at least two xUnit tests of a program you have written or found elsewhere. Attach here (1) the screenshot of your program execution and (2) the source code of your program.

```python
import pytest

def add(a, b):  1 usage
    return a + b

def divide(a, b):  1 usage
    try:
        return a / b
    except ZeroDivisionError:
        return "Cannot divide by zero"

@pytest.mark.parametrize("a, b, expected", [
    (5, 7, 12),
    (-4, 0.5, -3.5),
    (0, 0, 0),
    (-2, -3, -5),
    (1.5, 2.5, 4.0)
])

def test_add(a, b, expected):
    assert add(a, b) == expected

@pytest.mark.parametrize("a, b, expected", [
    (10, 2, 5),
    (-10, 2, -5),
    (0, 10, 0),
    (10, 0, "Cannot divide by zero")
])

def test_divide(a, b, expected):
    assert divide(a, b) == expected
```

```
============================= test session starts ==============================
collecting ... collected 9 items

sprint0/unit_test.py::test_add[5-7-12] PASSED                            [ 11%]
sprint0/unit_test.py::test_add[-4-0.5--3.5] PASSED                       [ 22%]
sprint0/unit_test.py::test_add[0-0-0] PASSED                             [ 33%]
sprint0/unit_test.py::test_add[-2--3--5] PASSED                          [ 44%]
sprint0/unit_test.py::test_add[1.5-2.5-4.0] PASSED                       [ 55%]
sprint0/unit_test.py::test_divide[10-2-5] PASSED                         [ 66%]
sprint0/unit_test.py::test_divide[-10-2--5] PASSED                       [ 77%]
sprint0/unit_test.py::test_divide[0-10-0] PASSED                         [ 88%]
sprint0/unit_test.py::test_divide[10-0-Cannot divide by zero] PASSED     [100%]

============================== 9 passed in 0.01s ===============================
```

3. GUI programming (4 points)

Write a GUI program in the language you have chosen for your SOS project. The GUI of your program must include text, lines, a check box, and radio buttons. While you are recommended to consider the GUI for the SOS game board, it is not required. In this assignment, any GUI program of your own work is acceptable.

Attach here (1) the screenshot of your program execution and (2) the source code of your program.


Sample SOS Game
Simple Game
Dynamic Game
Human Button
Computer Button

gui.py × unit_test.py

```python
import sys

from PySide6.QtGui import QPainter
from PySide6.QtWidgets import QLabel, QWidget, QCheckBox, QRadioButton, QGridLayout, QApplication

class Board(QWidget):  1 usage
    def __init__(self, rows = 3, cols = 3):
        super().__init__()
        self.rows = rows
        self.cols = cols


    def paintEvent(self, event):
        painter = QPainter(self)

        width = self.width()
        height = self.height()

        for colIndex in range(1, self.cols):
            xPosition = colIndex * width / self.cols
            painter.drawLine(int(xPosition), 0, int(xPosition), height)

        for rowIndex in range(1, self.rows):
            yPosition = rowIndex * height / self.rows
            painter.drawLine(0, int(yPosition), width, int(yPosition))

class Window(QWidget):  1 usage
    def __init__(self):
        super().__init__()

        label = QLabel("Sample SOS Game")
        checkboxSimple = QCheckBox("Simple Game")
        checkboxDynamic = QCheckBox("Dynamic Game")

        radioHuman = QRadioButton("Human Button")
        radioComputer = QRadioButton("Computer Button")

        board = Board(rows=3, cols=3)

        grid = QGridLayout()
        self.setLayout(grid)

        grid.addWidget(label, 0, 0, 1, 2)
        grid.addWidget(board, 1, 0, 1, 2)
        grid.addWidget(checkboxSimple, 2, 0)
        grid.addWidget(checkboxDynamic, 2, 1)
        grid.addWidget(radioHuman, 3, 0)
        grid.addWidget(radioComputer, 3, 1)

        self.resize(500, 500)
```