Fundamentals of OOP (5N0541)

Skills Demonstration 1

QQI Level 5

Issued: 22 Jan 2020 | **Submission Date:** 29 Jan 2021

Submission to include:

- Class Diagram
- ball.py file
- Completed test_ball.py file
- Evidence of testing (file or in report)

Assessment Brief

This assessment is worth 20%. It is a cross-modular assignment between Software Architecture (5N2772) and Fundamentals of OOP (5N0541).

READ ALL THE INSTRUCTIONS AND REVIEW THE MARKING SCHEME THOROUGHLY!

Instructions to Candidates

You will create a class "Ball" that implements several methods. The module name will be ball.py and you will be able to import it into any program that uses this shape. It will contain a number of methods associated with operator overloading and formatting of the data for the user.

You will also test the class (e.g. unit testing) using the starter code or your own generated code (such as doctest tests) to ensure that all methods including overloaded methods will work correctly.

- Code and test an object oriented program that provides a solution to the problem statement that:
  - contains a single class
  - features appropriate data types to simulate and/or model the problem
  - correctly performs type conversions

You will submit evidence including (but not limited to):

- appropriate program documentation to include relevant screen captures, digital/visual evidence of debugging, testing, etc.) and indicative critical reasoning
- Python file for the module (the class)
- Python file for the Test module (unit test or self-designed test file)

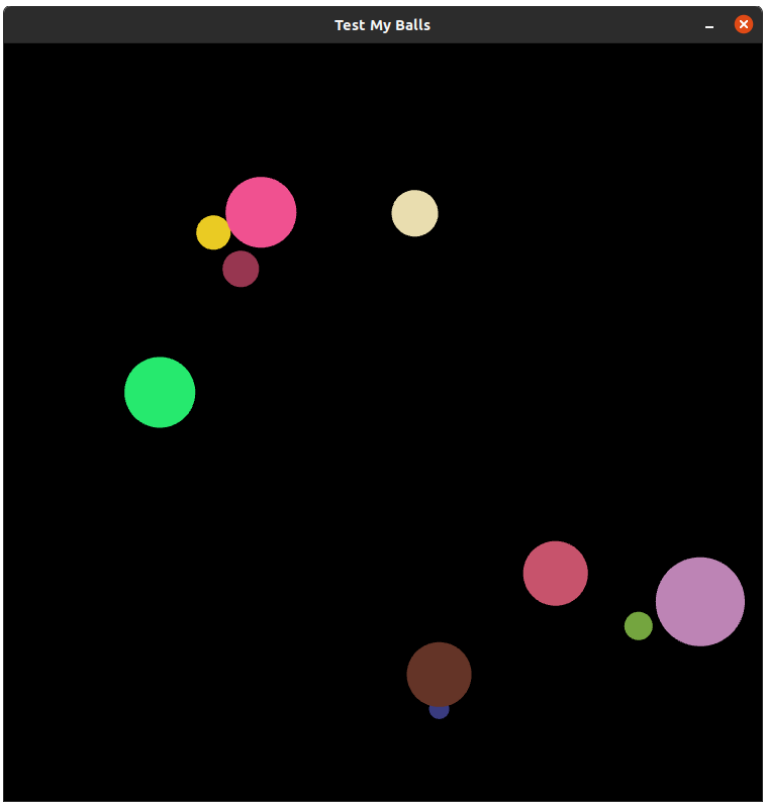
Marking Scheme for class Ball	Max Mark
1) Clearly Documented Source Code a. Create the class diagram for the Ball class. (10) b. The class file has appropriate header information, the methods include complete documentation (docstrings) (10)	**20**
2) Program Functionality: a. The init ensures that only valid args are accepted to create an instance. If a call is made with a positive value less than 1, this will mod the radius to 1 and not raise an error, but a value of 0 or less, or a value of the wrong type will raise an error. (8) b. Bad values or incorrect types being used in overloaded methods are rejected and raise an appropriate error and message. (6) c. Code is DRY - used helper functions/methods where appropriate (6)	**20**
3) Accurate programming (syntactically and semantically): a. Class constructor has default value(s) and well-named attributes for **radius** (min val of 1), random **color** (from 30-255) and random (x,y) **pos** (100-700). (6) b. The overloaded repr is implemented as shown in sample data. (2) c. The area (2), volume (2), set_pos (1) and set_color (1) methods are implemented d. The overloaded arithmetic (+ - / *) methods are implemented accurately. The arithmetic operators will use volume as the criteria to add, sub, mul and div. Instances of add and sub use two ball objects, but when used with mul and div the other value will be a positive number. (12) e. The new Ball produced from arithmetic ops will have the position of the original (2) f. The overloaded boolean (==, !=, >, <, >=, <=) methods are implemented and are accurate (6) g. The draw function works correctly (6)	**40**
4) Appropriate Testing and Debug a. Complete the code to create 10 instances with the last two having same radii (2) b. Complete the code to change the colors, and show before and after (2) c. Complete the code to run tests for the logical operators using the methods shown for the arithmetic operators. (5) d. **Show evidence** that you have tested each of the methods specified in the terminal using the output from the tests (8), and a screenshot with balls showing on the screen. (3)	**20**

Screenshots of tests(partial)

```
                TESTS FOR BALL CLASS

---Instantiation and repr tests---

        All instantiations successful


        These are the objs to be used in all tests

         Ball obj: radius 17; color (234, 203, 35); pos (612, 179)
         Ball obj: radius 18; color (151, 54, 80); pos (358, 224)
         Ball obj: radius 35; color (240, 81, 144); pos (535, 117)
         Ball obj: radius 44; color (189, 132, 181); pos (192, 508)
         Ball obj: radius 10; color (57, 59, 127); pos (589, 654)
         Ball obj: radius 35; color (38, 233, 110); pos (327, 695)
         Ball obj: radius 23; color (233, 221, 175); pos (173, 366)
         Ball obj: radius 14; color (116, 165, 63); pos (227, 302)
         Ball obj: radius 32; color (199, 83, 108); pos (419, 652)
         Ball obj: radius 32; color (100, 52, 39); pos (323, 512)
        Note: All instantiations printed using __repr__ method



-- Testing area_tests method --

        Ball obj: radius 17; color (234, 203, 35); pos (612, 179) has area 3631.681107549801
        Ball obj: radius 18; color (151, 54, 80); pos (358, 224) has area 4071.5040790523717
        Ball obj: radius 35; color (240, 81, 144); pos (535, 117) has area 15393.804002589986
```

Initial tests for instantiations and repr method

You could take the entire output of the tests and place into a text file to submit with the python files, as evidence of all tests being run, and the result of all tests.



Drawing tests run successfully

Include a snip of the ten instances created being drawn to the screen using the function provided.