CSE-165 Lab 09

Points: 100

Write a separate file for each of the following tasks.

Zip all your files together and submit the zip file to CatCourses.

---

1. **Copy Constructors (20 points)**

Inspect the file objects.cpp. It creates instances of the Object class and prints the value of the count variable, which is a static variable that keeps track of how many instances of Object have been created.

Your task is to implement the Object class. It only needs to have a static member variable named count, which should be incremented each time a new instance of the class is created. You should provide the appropriate constructors. If you are getting a count other than 3 from objects.cpp, then you are probably missing one. (Do not worry about destructors for this exercise.)

Save your class in a file named Object.h.

Expected Output:

3

---

2. **Copy Constructors II (10 points)**

In the previous exercise, there were 3 instances of the Object class being created. One of those was being created because we passed the object into the function f by value, creating a copy. Modify the function f so that this does not happen. Your program should report that there are 2 objects, not 3. Upload your modified version of objects.cpp.

Expected Output:

3

CSE-165 Lab 09

Points: 100

3. **Dynamic Sortable (50 points)**

Reuse your Sortable, Circle, Participant, and Data classes from Lab 8. (You may use the simpler version of Participant from exercise 4 of lab 8, instead of exercise 5.)

Your task is to make all of your classes work correctly with sortingData.cpp. Note that sortingData.cpp creates a mixture of both Circles and Participants; this means that your compare methods cannot assume they can always simply downcast Sortable pointers to their own type. You will have to use dynamic_casts and/or the typeid function to check what kind of object the compare method receives and act accordingly. Refer to the last lecture slides on Polymorphism.

When comparing two objects of the same type, use the same logic as in lab 8. But when asked to compare a Circle with a Participant or vice-versa, by definition we will consider a Circle < Participant, so that, after sorting, all Circles should show up in the Data vector before all of the Participants. (See the expected output below.)

Submit your modified Circle and Participant classes.

Expected Output:

```
Waymond 24  100
Circle with radius: 0
Mary    27  96
Circle with radius: 3
John    32  100
Circle with radius: 2
Eliza   21  105
Circle with radius: 4
Ezekiel 27  96
Circle with radius: 1
Alex    20  101

Circle with radius: 0
Circle with radius: 1
Circle with radius: 2
Circle with radius: 3
Circle with radius: 4
Eliza   21  105
Alex    20  101
Waymond 24  100
```

```
John    32  100
Ezekiel 27  96
Mary    27  96
```

---

4. **Dynamic Sortable II (20 points)**

Reuse your Sortable, Circle, Participant, and Data classes once more.

Modify sortingData.cpp so that instead of a fixed number of hard-coded Circles and Participants being created and added to Data, the user instead has control of manipulating the Data object via input.

Create a while loop where you continuously ask the user for their choice of action, with at least the following options available:

'q' or 'Q' - break out of the loop and end the program

'c' or 'C' - add a new Circle to Data (let the user also decide on the radius of the circle)

't' or 'T' - add a new Participant to Data (let the user also decide on the participant's information)

's' or 'S' - sort the Data

'p' or 'P' - print the Data

(You may choose different characters for each action, as long as the actions themselves are available.)

Submit your modified sortingData.cpp.

Example Run:

```
c 10
Circled added
c 5
Circled added
t John 20 95
Participant added
c 7
Circled added
t Jane 25 100
Participant added
```

CSE-165 Lab 09

Points: 100

```
c 12
Circled added
t Xavier 34 102
Participant added
t Anna 32 102
Participant added
p
Circle with radius: 10
Circle with radius: 5
John    20  95
Circle with radius: 7
Jane    25  100
Circle with radius: 12
Xavier  34  102
Anna    32  102
s
Data sorted
p
Circle with radius: 5
Circle with radius: 7
Circle with radius: 10
Circle with radius: 12
Anna    32  102
Xavier  34  102
Jane    25  100
John    20  95
q
```