Classes & Objects

Complete this worksheet by answering the questions using the given program. Make sure to color or **bold** your answers so they are distinguishable from the question.

Getting Started

- ☑ ~~Make a copy of this document by selecting "Make a Copy" from the File menu.~~
- ☑ ~~Go to https://replit.com/~ and log in with gmail using your Weber State email address.~~
- ☑ ~~Open the code linked below~~
- ☑ ~~Click **fork**, located in the upper right hand corner, to create your own copy~~
- ☑ ~~In your copy, go to the upper right hand corner, click **invite**. Then choose **Generate a Join Link**.~~
- ☑ ~~Paste the generated link in the second bullet point below~~

Worksheet Program Links

- *Starter Code*: Repl Classes & Objects
- *Finished Program Code*: https://replit.com/join/wedbulhwfc-trinanixon1

Part 1 - Class Setup

1. Observe the Distance class. What file is the class specification declared in? **Distance.h**

2. What file are the Distance class member functions defined in? **Distance.cpp**

3. Notice that the Distance.cpp file has the following preprocessor command listed at the top of the page:

   ```
   #include "Distance.h"
   ```

   What is the purpose of this command? **It will include the external file 'distance.h', which is the header file that provides all the declarations for the distance class. Without it, the program would likely have errors when it is ran (e.g. there are undeclared identifiers, etc);**

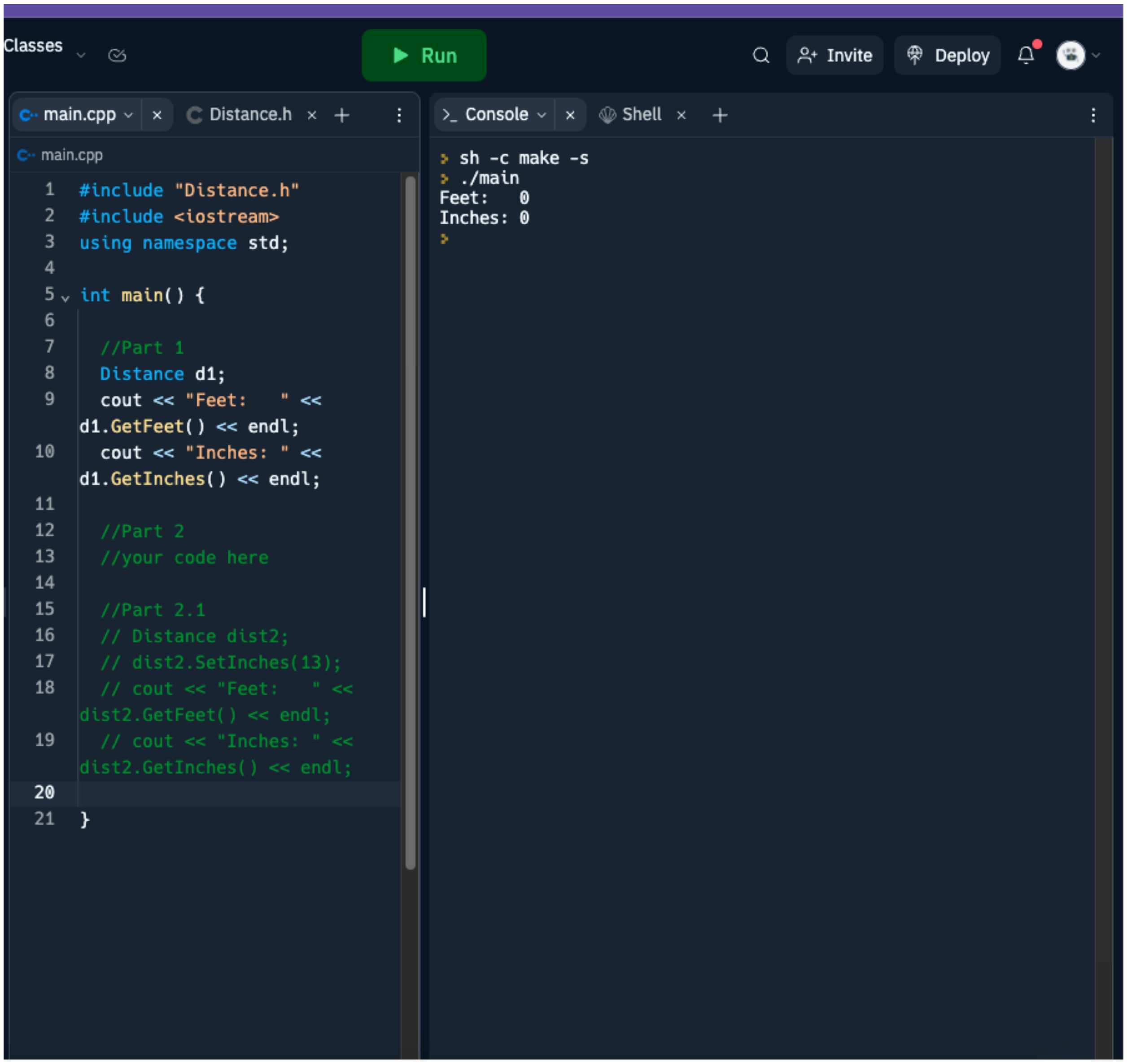
4. How does each function defined in the .cpp file know that it belongs to the Distance class? **The scope operator is used(::) to access the variables, functions, or types that belong to the Distance class.**

5. Observe the Distance.h file, which class members are designated as private? Why are they designated that way? **Double feet_; and double inches_; are both designated as private. Designating these members as private means that these variables are only accessible and only modified from within the Distance class.**

6. Which class members are designated as public? Why are they designated that way? **The functions SetFeet() and SetInches() are designated as public. Also, GetFeet() and GetInches() are public. These being set as public allows for controlled access to the members of the Distance class. So a user of the Distance class can work with and modify the objects, as needed.**
7. In the Distance class, change the word **private** to **public**. Look at the main.cpp file under Part 1. Run the program. Does it compile? **It does compile. It shows 0 for both feet and inches**

8. Go back and change the **public** designation to **private** again (above the member variables). Now try to run the program again. What happens?
   We get 2 errors. It does not compile.
9. Fix the program so that it will print the member variables using the class's public interface. Hint: use the GetFeet() and GetInches() functions.

10. Put a screenshot of the working output below.



```
#include "Distance.h"
#include <iostream>
using namespace std;

int main() {

  //Part 1
  Distance d1;
  cout << "Feet:   " <<
d1.GetFeet() << endl;
  cout << "Inches: " <<
d1.GetInches() << endl;

  //Part 2
  //your code here

  //Part 2.1
  // Distance dist2;
  // dist2.SetInches(13);
  // cout << "Feet:   " <<
dist2.GetFeet() << endl;
  // cout << "Inches: " <<
dist2.GetInches() << endl;

}
```

```
> sh -c make -s
> ./main
Feet:   0
Inches: 0
>
```

Part 2 - Constructors

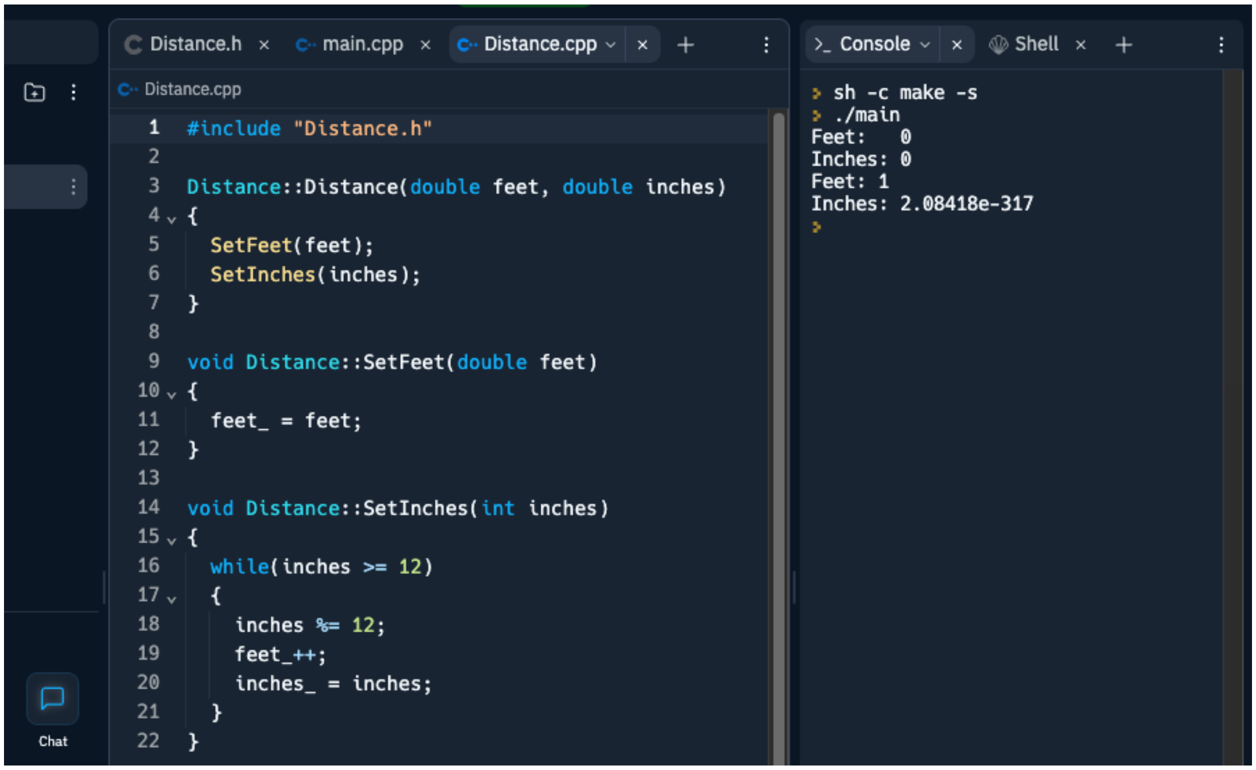
1. How many constructors are present in the Distance class? 2 constructors are present. Default Constructor - initializes feet_ and inches_. There's a constructor Distance(double feet, double inches). This initializes feet and inches with provided values vs the default which initializes to zero by default when no initial values are provided.

2. In part 1 there is a variable called d1. Which constructor is used to create this object? The default constructor Distance() is called without any argument.

3. What values are given to d1's member variables, why? by default, the d1 objects feet_ and inches_ member variables are initialized to zero. The default value is zero when no specific value is given.

4. The default constructor is defined using an initializer list. Observe the values added to the () next to each member variable in the list. What are these values?
These values are the initial values assigned to those variables when the object is created. So in this case, feet_ is initialized to 0, and so is inches.

5. Now observe the curly braces at the end of the list. Are there any values in the curly braces? Why?
No, there are no values in the curly braces. The purpose of the default constructor is to initialize member variables, and since this is already handled by the initializer list in the default constructor, there is no need for any additional code in the curly braces. No additional actions beyond initialization are required by the default constructor.
6. Observe the general constructor. Where is the prototype for it? Where is it defined?
On line 12, the general constructor is as follows: Distance(double feet, double inches);
This is the prototype. The Distance.cpp file is where this general constructor is defined - in the SetFeet() and setInches() functions.
7. In main, under the Part 2 comment, create a Distance object called d2 that has 1 foot and 11 inches using the general constructor. Write code to print the member variables and run the program. Paste a screenshot of the output here.



```
#include "Distance.h"

Distance::Distance(double feet, double inches)
{
  SetFeet(feet);
  SetInches(inches);
}

void Distance::SetFeet(double feet)
{
  feet_ = feet;
}

void Distance::SetInches(int inches)
{
  while(inches >= 12)
  {
    inches %= 12;
    feet_++;
    inches_ = inches;
  }
}
```

```
> sh -c make -s
> ./main
Feet:   0
Inches: 0
Feet: 1
Inches: 2.08418e-317
>
```

8. Now create another object called d3. This time give it 1 foot and 15 inches. Write code to print the output. Does it display 1 foot and 15 inches? Why or why not. It displays Feet: 2 and Inches: 3. Which is technically the same as 1 foot 15 inches. The reason it calculates it this way can be found in the SetInches() function. When inches is greater than or equal 12, it enters the loop and 12 inches is converted to 1 foot. (feet++ increments by 1). The value of inches after the loop will be 3. The loop will not execute again because 3 < 12. So inches will be assigned 3.

9. Why do we use SetInches and SetFeet in the general constructor's definition to set our member variables instead of simply setting the member variables equal to the parameters? By using the setter functions as opposed to just directly setting the variables provides better control of the data within the class. We can ensure data integrity and better maintain our code.
10. Uncomment the code under Part 2.1. What kind of constructor is used? What will the output be?
    A default constructor is being used: Distance dist2; The output will be Feet: 1 and Inches: 1.
11. Run the code. What is the output? Were you correct? Yes, I was correct. The output is Feet: 1 Inches: 1