Contents

1 01.21.20 (C++ Ch. 3)

1.1 Objects and Object Sizes

- An objects size will always be the sum of its data members. The size will not be affected by any methods that are called upon it.
- Because of this, objects can quickly become very large in size.

1.2 UML Diagrams

- Classes are listed as individual boxes
  - top box = class name
  - middle compartment =

1.3 Constructors

- Explicit constructors can be used to prevent implicit typecasting, as seen below:

```
class Student {
  Student (int s) {

  } //constructor
} //Student

int main () {
  Student s {15}; //allowed, completes correctly
  Student c {'C'}; //typecasts automatically, should not occur
  //Note, () can be used in place of {} to construct objects
}
```

- Ex. list initialization with an explicit constructor

```
explicit Account (std::string accountName) //explicit constructor
  : name{accountName} {
  //insert constructor code here
  }
```

2 01.19.20 (C++ Ch. 3)

A look at class creation

```
#include <iostream>
using namespace std;

//defining the class
class GradeBook {
  //holds all public vars, functions
  public:
  //public function
  void displayMessage() {
    cout << "Welcome to your Gradebook" << endl;
  } //displayMesage
} //GradeBook

//main method
int main () {
  //creates a GradeBook object
  GradeBook myGradeBook;
  //calls above created function on object
  myGradeBook.displayMessage();
}
```

- Class functions and vars are, by default, private. The public keyword must be used to denote any public parts of a class.
- Move implementations to a header file for use in main methods while separating out each file.
- When using header files, use quotation marks around them to indicate that they're a file on your machine. Use angle brackets around things to include form the C std lib.
- The purpose of const functions is to prevent the function from modifying the values of data members or objects.

3 01.19.20 (C++ Ch. 2)

A look at some basic C++ code

```
#include <iostream> //enables program to output data

//main function begins program execution
int main () {
  //cout currently a function as a part of the std namespace
  std::cout << "Welcome to C++!\n";
  //above << is an insertion operator, overloaded from the bitwise left-shift

  return 0;
}
```

A look at some higher level C++ code

```
#include <iostream>

int main () {

  int num1{0}; //list initialization
  int num2 = 0; //regular initialization
 //No difference between list & regular initializtion with primitive types.
 //List initialization should be used for UDTs.

  int sum{0}

  std::cin >> num1;
  std::cin >> num2;

  sum = num1 + num2;

  std::cout << sum << std::endl;
  //endl is helpful because it flushes the buffer
  //newline character does not
  return 0;
}
```

A look at a common mistake

```
#include <iostream>

int main () {
  int x {5};
```

```
  if(x > 10); {
    std::cout << x "> 10" << std::endl;
  }
  //still prints output because of semicolon after if statement

  return 0;
}
```