THE UNIVERSITY OF WESTERN ONTARIO
LONDON CANADA

COMPUTER SCIENCE 3307A
FINAL EXAMINATION
DECEMBER 13, 2018
3 HOURS

NAME: Ali Al-Musawi

STUDENT NUMBER: _____

Question	
1-20.	_____
21-22.	_____
23.	_____
24.	_____
25.	_____
26.	_____
27.	_____
28.	_____
29.	_____
30.	_____
31.	_____
32.	_____
33.	_____
34.	_____
35.	_____
36.	_____
TOTAL	_____

(Out of 180 marks)

There are no cheat sheets, books, or other reference materials allowed for this exam. No calculators or other electronic devices are permitted either.

Part I -- Multiple Choice, True/False -- Choose the best answer from the choices given. Circle your answer on the paper. [40 marks total, 2 marks each]

1. The following is (are) true about C++.
   a. It allows the use of objects.
   b. It allocates all objects on the heap.
   c. Programs run using an interpreter.
   d. It performs garbage collection.
   e. More than one of the above statements are true.

2. Following standard conventions, a .h file for a C++ class must contain:
   a. The class declaration. ✓
   b. A `#ifndef` statement. ✓
   c. A `#include` statement.
   d. Two of the above are true.
   e. All of the above are true.

3. When writing C++ code, the programmer and not the compiler is responsible for:

   1. Ensuring no uninitialized values are used. ✓
   2. Ensuring there is the correct amount of space allocated for each stack frame.
   3. Ensuring memory allocated on the heap is reclaimed. ✓
   4. Ensuring pointer addresses used are correct. ✓

   Which of the following combinations of statements from the above list is correct:
   a. Statements 1 and 4 are correct.
   b. Statement 2 is correct.
   c. Statements 1, 3, and 4 are correct.
   d. Statements 2, 3, and 4 are correct.
   e. None of the above options are correct combinations.

4. A scope is the part of a program in which a name (of a variable or function, for example) is visible.
   a. True.
   b. False.

5. A class in C++ may only have one destructor.
   a. True.
   b. False.

6. A constructor in C++ may be overloaded.
   a. True.
   b. False.

7. What is the output of the following program?

```
#include <iostream>
int f(int n, int & v, int * p) {
   v = *p;
   v = v + 1;
   return n+(*p);
}

int main() {
   int n = 10;
   int m = 20;
   std::cout << f(n, n, & m);
}
```

a. 30.
b. 31.
c. 41.
d. 42.
e. An error occurs.

8. What is the output of the following program? Note that both the signature and the body of the function `f` have changed from the previous question.

```
#include <iostream>
int f(int n, int v, int * p) {
   *p = v;
   v = v + 1;
   return n+(*p);
}

int main() {
   int n = 10;
   int m = 20;
   std::cout << f(n, n, & m);
}
```

a. 20.
b. 21.
c. 31.
d. 41.
e. An error occurs.

9. The declaration:

```
A *a = new B;
```

a. Will compile only if `A` is a subclass of, or equal to, `B`.
b. Will compile only if `B` is a subclass of, or equal to, `A`.
c. Will always compile.
d. Will not compile unless `A` and `B` are of the same type.
e. None of the above.

10. Consider the following program:

```
class Base {
   public:
      void foo(int) {};
};

class Derived : public Base {
   public:
      void foo(int) {};
};

int main () {
   Derived* d = new Derived;
   d->foo(42);                    // The question is about this line
}
```

Which version of `foo` is executed on the line commented above?

a. `Base::foo`.
b. `Derived::foo`.
c. None of the above: it results in a compilation error.
d. None of the above: it results in a run-time error.

11. Assume that the class `Card` is well defined and is available to the following class declaration.

```
class Deck {
   public:
      void shuffle();
      Card *dealTopCard();
   private:
      Card *deck_[52];
};
```

The default destructor for `Deck` will not free the memory pointed to by the `Card` pointers stored in `Deck`'s `deck_` data member.

a. True.
b. False.

12. Assume that the class `MyClass` is well defined, has no memory leaks, and is available to the following `main` function.

```
int main() {
   MyClass c;
   MyClass array[5];
   MyClass *ptr;
}
```

Is there a memory leak in this program?

a. Yes.
b. No.

13. Continuing the above example, how many times is the default constructor of `MyClass` called when the `main` function is executed?

a. 0.
b. 1.
c. 6.
d. 7.
e. None of the above.

14. Continuing the above example, how many times is the default destructor of `MyClass` called when the `main` function is executed?

a. 0.
b. 1.
c. 6.
d. 7.
e. None of the above.

15. Continuing the above example, memory for `array` is allocated:

a. Off of the stack.
b. Off of the heap.
c. In the global memory space.
d. No memory is allocated.
e. None of the above.

16. The story points of a user story:

a. Have a work-hours equivalent defined specifically in the user story specifications.
b. Cannot change once set.
c. Are used to indicate the approximate size and complexity of a story.
d. Are estimated by the customer.
e. None of the above.

17. You should use this design pattern when extension by subclassing is impractical and you need to add responsibilities to individual objects dynamically:
    a. Composite.
    b. Visitor.
    c. Decorator.
    d. Singleton.
    e. None of the above.

18. The Factory Method design pattern is useful for:
    a. Encapsulating the construction of a concrete object.
    b. Constructing decorated objects with the right decorators.
    c. Constructing the appropriate iterator for a composite object.
    d. All of the above.
    e. None of the above.

19. Name the design pattern used in the following example:

```
class SearchFactory {
   private:
     static SearchFactory *searchFactory;

   protected:
     SearchFactory() {
     }

   public:
     static SearchFactory *getSearchFactory() {
       if (searchFactory == NULL)
         searchFactory = new SearchFactory();
       return searchFactory;
     }
     SearchDialog *getSearchDialog(){
       return new GoToDialog();
     }
};
```

    a. Singleton.
    b. Factory Method.
    c. Abstract Factory.
    d. Composite.
    e. None of the above.

20. The Builder pattern is used to:
    a. Create objects by merging classes with each other.
    b. Control the structure of complex, hierarchical class relationships, by building up from the bottom.
    c. Control the creation of objects where there are many constructors with long and complex parameter lists.
    d. Create objects through a similar process of steps, but allow the actual representation to vary.
    e. None of the above.

Part II -- Fill In The Blank -- Indicate any compile-time or run-time problems with the following C++ code. Show output where appropriate. Write "ok" if there are no problems with the line in question. Note: simply writing "error" for a given line will not earn you any marks; you must indicate the nature of the error as well. [23 marks total]

21. Concerning `const` and pointers [12 marks total, 1 mark per line]:

```
1  int* a = new int [10];
2  const int* b = a;
3  a[0] = 10;
4  b[1] = 20;
5  int* c = b;
6  int* const d = a;
7  const int* e = b;
8  int* const f;
9  d[3] = 50;
10 d = a;
11 d = b;
12 int* g = d;
```

Line 1 ok

Line 2 ok

Line 3 ok

Line 4 Compile-time error

Line 5 compile-time error

Line 6 ok

Line 7 ok

Line 8 compile-time error

Line 9 ok

Line 10 Compile-time error

Line 11 compile-time error

Line 12 ok

22. Concerning references [11 marks total, 1 mark per line]:

```
1  int i = 10;
2  int& j = i;
3  int& k;
4  int& const l = i;
5
6  j = 20;
7  cout << i << endl;
8
9  const int& p = i;
10 p = 30;
11 i = 30;
12
13 const int& q = j;
14 cout << q << endl;
```

Line 1 ok

Line 2 ok

Line 3 compile-time error

Line 4 compile-time error

Line 6 ok

Line 7 ok (20)

Line 9 ok

Line 10 compile-time error

Line 11 ok

Line 13 ok

Line 14 ok (30)

Part III -- Short Answer -- For the following questions, write your answer in the space provided, using diagrams as appropriate. You do not need to write full sentences, and can use point form if that is your preference. [60 marks total]

23. Assume that the class `Balloon` has a public constructor that takes a single string (`std::string`) as an argument. The code fragment below will cause an error. Explain which line causes an error and why. [6 marks]

```
Balloon b("Red");
Balloon *ptr = &b;
delete ptr;
```

The last line causes a runtime error because b does not have memory allocated on the heap, yet delete ptr attempts to reclaim already-free memory since memory for b is allocated on the stack. Only pointers to heap memory can be deleted.

24. In C++, what is the difference between a virtual method and a non-virtual method? Why is there a distinction between them in C++? [6 marks]

The difference allows for dynamic dispatch. When a derived class overrides a base class method, and a derived object on the heap is referenced with a base pointer then :

- If the method is virtual, invoking the method on the derived object will cause the overridden method to be called.
- If the method is non-virtual, invoking the method on the derived object will cause the base class method to be called.

25. How would you declare a class `A` to prohibit the copying of objects of type `A`? (You may provide sample code or an explanation to answer this.) [6 marks]

Ans 1: One can make the copy constructor private, however that means the class itself can always access the copy constructor. To completely prohibit copying, all methods in that class should pass instances of the same class by reference. In addition, new instances should not be returned by value. This is because once the method is done, the new object is copied into the stack memory of the caller. For instance, if the class is named A, then we should never use the return type A and should never use a function/method parameter of type A.

Ans 2: C++ allows deleting the copy constructor so that a compiler error is thrown whenever one attempts to copy explicitly or implicitly. This is done by declaring the copy constructor and assigning "delete" to it. For example, with class A:

```
A(const A& a) = delete;
```

26. For each of the following user stories, indicate whether or not the story is acceptable according to the INVEST criteria we studied in class. If not, indicate which criteria the story violates. Briefly justify your answer. [6 marks total]

a. *All connections to the database are through a connection pool.* [3 marks]

Not acceptable → violates the "Valuable" criterion as the customer/client may not understand the technical jargon used here.

b. *A job seeker can find a job.* [3 marks]

Not acceptable → violates the "Small" criterion. This user story can be used as an epic and get decomposed into smaller user stories. For instance:

- As a user, I can search for jobs by business domain.
- As a user, I can search for jobs by category (part-time, full-time, contract, internship).
- As a user, I can track my job applications.
- As an employer, I can make a job posting.

27. C++ allows one to pass variables by-value, by-pointer, and by-reference. Describe the similarities and differences between each of these methods. [6 marks]

By pointer and By reference allow mutating the object they are associated with in the function (if the object is not const). By value does not allow this since a copy is passed and destroyed when the function terminates. By-pointer and By-reference are computationally negligable compared to By-value as only the address is passed in the former. By-pointer is more flexible than by-reference since the pointer can be re-pointed (as long as it is not a const pointer) while the reference is const and cannot be re-assigned: i.e. the reference cannot be made to refer to another object.

28. List three particular reasons why it is useful to be able to inherit characteristics from parent classes in object-oriented programming. [6 marks] Dumb Question :)

1. Code Re-use: no need to define the same function over and over again.

2. Ease of Varying Implementation: changing the implementation can be done once only and all extending (derived) classes get the new implementation.

3. Code Readability: Derived classes focus only on adding/overriding functionality, making it easier to infer/contrast/compare against the base class.

29. Briefly explain the main problem, if any, with the following code, assuming that the class `Person` has been properly defined. If this code will execute correctly, explain why. [6 marks]

```
#include "Person.h"

Person *f() {
  Person p("Joe");
  return &p;
}
```

The function is returning a pointer to nothing since p is created on the frame stack of f. Once f returns, a pointer to p is returned, then p is destroyed. When the returned pointer is de-referenced outside of f, a segmentation fault will occur.

30. Explain why low coupling and high cohesion often go hand-in-hand. [4 marks]

These are characteristics of well-designed modular software. When a component/class has low cohesion, we can decompose it into different units with low coupling, each of which has high cohesion. On the other hand, when we have highly coupled components, we can merge some of them and clean up their interfaces. The result does not degrade cohesiveness but may improve it. In addition, we improve coupling by lowering it.

31. Discuss the advantages and disadvantages associated with the use of accessor methods (getter and setter methods for individual attributes) to access the state of an object indirectly. Can the disadvantages be addressed in C++, and if so, how? [8 marks]

Advantage	Disadvantage
• The implementation can always be changed compared to making the data members public. • Useful for derived classes when private inheritance is used.	• Allow external classes to change the state of the object even if for some objects this is not desired. • Defeats the point of information-hiding.

The disadvantages can be addressed by labelling the object whose state should not be modified by the key word const. In addition, not all attributes have to be exposed, especially if some are implementation details.

32. Why should classes be designed so that they can be fully initialized by their constructors? (If they aren't, what impact does it have on their public interface?) [6 marks]

Classes should be fully-initialized by their constructors so that we don't impose an order on which method of the pubic interface should be called first. For instance, if some of members are not initialized as intended for their use, then calling a method that operates on the un-initialized data member may return invalid result or throw an exception. This forces an order on how to use the class, i.e. call setters first.

Part IV -- Long Answer -- For the following questions, write your answer in the space provided, using diagrams as appropriate. Again, you do not need to write full sentences, and can use point form if that is your preference. [57 marks total]

33. Recall the system utilities created in the individual assignment earlier in this course. This assignment entailed the creation of a number of classes to retrieve various bits of system information, along with utilities to present this information individually (as command line utilities) and through a text-based menu interface. [20 marks total]

a. Craft a user story capturing one of the requirements for this assignment. Use the front-of-card and back-of-card areas below to capture and record all of the requisite information for this user story. [8 marks]

Front of Card
See my 2019 Solution

Back of Card
See my 2019 solution

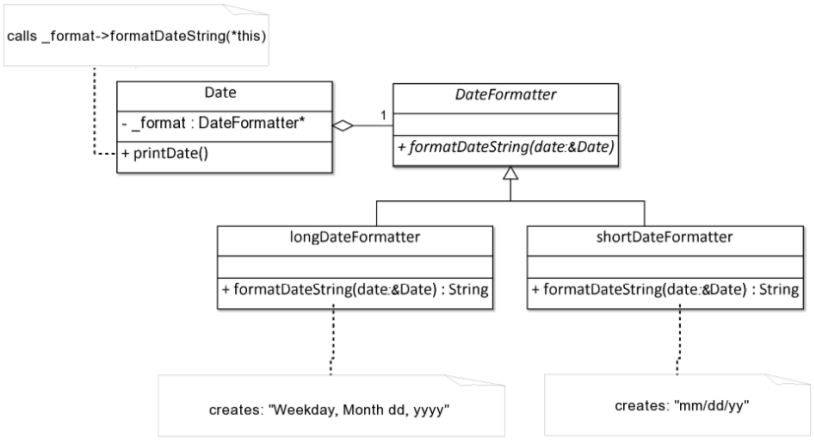
b. Select one of the creational, behavioural, or structural design patterns discussed in class and discuss how you could apply this pattern to this assignment. In doing so, indicate how the use of this pattern would improve the design and quality of this application. [8 marks]

see my 2019 solution

c. Provide a UML diagram, like those given in class accompanying our design pattern examples, to illustrate the use of and integration of your selected pattern from part b) above into the assignment. [4 marks]

see my 2019 solution

34. Consider the following class diagram. [14 marks total]



a. Identify one design principle from class that this system adheres to and explain briefly why/how it accomplishes this. State two benefits of using this design principle. [8 marks]

Encapsulate What Varies. Since the date format can be specified differently regardless of the date, it makes sense to vary formatting using a class of its own and getting the Date class to own a formatter using aggregation

Benefits:
- The DateFormatter can be changed independently of the Date class.
- It is easy to add new types of DateFormatter by simply subclassing the DateFormatter, thereby achieving low coupling.

b. Which design pattern discussed in class is represented in this system? Justify your answer. [6 marks]

Strategy

- Solves the same problem as the strategy DP: Formatting a Date (i.e. Date to string) is an objective that can be solved by different algorithms. Each DateFormatter subclass is one algorithm.

- Uses the same UML template. The context owns a strategy interface that is implemented by different concrete strategies. A better implementation of this pattern allows changing the strategy without having to re-instantiate the Date object.

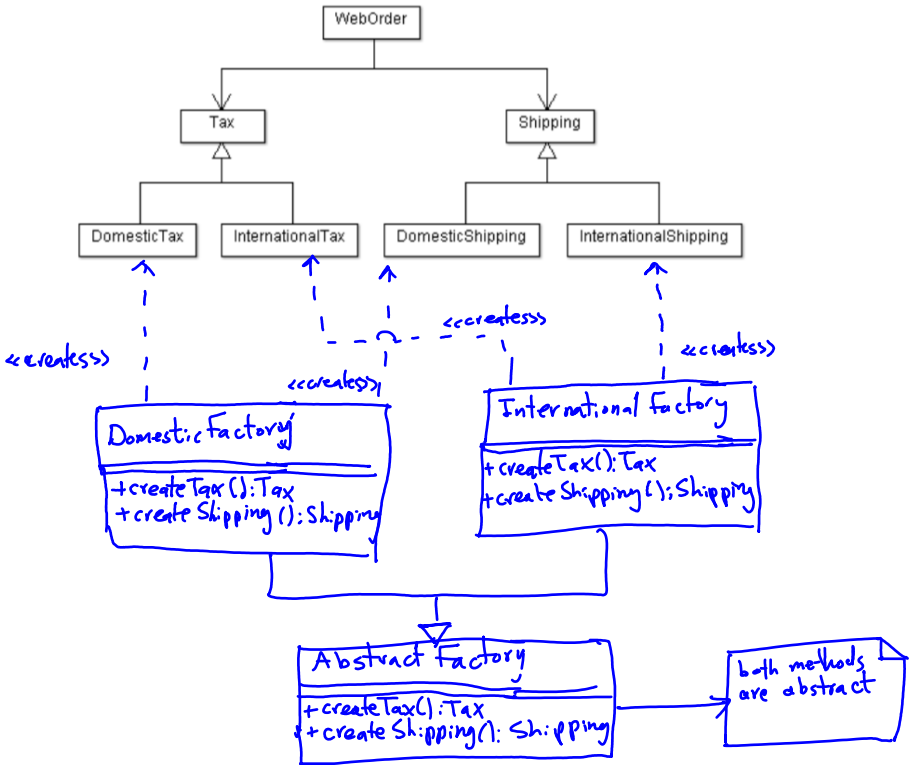
35. Software for the back-end of a web server which sells products to people worldwide is under development. For each order, there is some common information, but the details for calculating taxes and shipping in certain situations (like whether the order is Domestic or International) are different enough that the base classes Tax and Shipping are abstract, and subclasses have been created to deal with each situation. In WebOrder, the situation (Domestic or International) is available, and for each, instances of Tax and Shipping appropriate for that situation should be created. ie. If it is a Domestic order, DomesticTax and DomesticShipping subclasses for the WebOrder's Tax and Shipping attributes should be created.

    The WebOrder class should be as generic as possible, so that different tax and related shipping types could be added as subclasses in the future, without making changes to the WebOrder class. [13 marks]

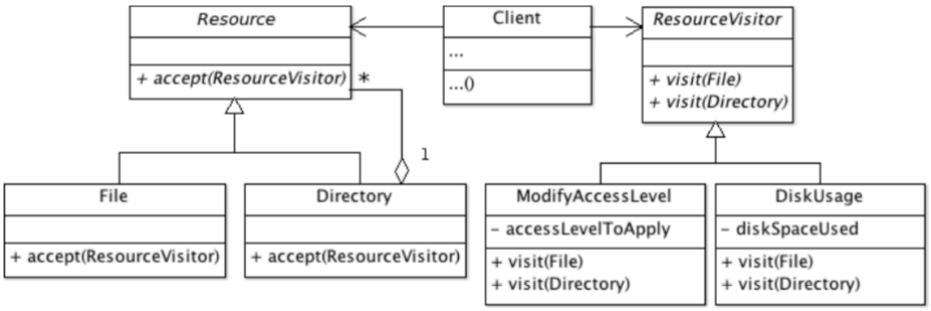
    a. What creational design pattern would you suggest for this situation and why? [3 marks]

    Abstract Factory because Domestic and International are good candidates for concrete factories and they both instantiate related "products"; in this case: Tax and Shipping. Using this pattern also accomplishes the generic constraint since we differ which objects to create to the subclasses (concrete factories).

    b. Add to the UML class diagram below to show how you would implement your design pattern choice from part a) above. Feel free to add notes or descriptions to clarify your design as necessary. [10 marks]



36. Consider the class diagram below. [10 marks]



a. Identify the design patterns used in the above system and identify their types (creational, behavioural, or structural). [2 marks]

- Composite : Structural
- Visitor : Behavioural

b. Describe one advantage of having applied either of the two design patterns to this context. The advantage must be a specific advantage of using that pattern, rather than a general benefit of using a design pattern. [4 marks]

- Composite: File-systems are inherently hierarchical, i.e. directories can be nested in each other, and can be empty or contain files. This structure is directly mapped and represented by the classes of the composite design pattern:

Resource ← [File ⇄ Leaf, Directory ⇄ Composite] → Component

c. Summarize what you believe the purpose the system is, and how the design patterns are used to implement this. [4 marks]

The purpose of this system is to be able to iterate over a resource (file or directory) and all of its children and be able to change their access permissions and collect/print statistics about their disk space usage as they are visited. The client can iterate over each resource, and while each resource is not a file, iterate over the resources children. If the resource is a file, perform the needed action by passing the appropriate visitor to the resource so it can accept it and perform the needed action (whether this is printing disk usage or changing access level). Note the same can be done to directories. These visitors roughly correspond to ls -l and chmod -r in linux.

This page has been left intentionally blank. Use it as additional workspace or extra space for answers if necessary.