

CPSC 5011 – Object-Oriented Concepts
Final Exam – Practice
Tuesday, December 5, 2017, 5:30-8:00pm

Name: _____

Circle One: Section 1 (Tue, 6:00-8:40pm) Section 2 (Thu, 3:45-6:25pm)

Instructions

- The exam is to be completed individually. No collaboration.
- This is a closed-book and closed-notes exam.
- Electronic devices (including calculators) are not allowed.
- You will have 2 hours and 30 minutes to complete the exam.

Comments

- Make sure your answers are legible.
 - If it is unreadable, it is incorrect.
- **SHOW YOUR WORK!**
 - Correct answers without work may be penalized.
 - Partial credit is possible if work is shown.
- For code construction portion(s),
 - Give the needed code and all necessary structures (prototypes, variable declarations, etc.)
 - Do not comment extensively or give elaborate I/O messages.
 - You may hardcode on this exam.

Question	Points	Your Score
1	12	
2a	18	
2b	16	
3	20	
4	10	
5	10	
6	10	
7	10	
8	40	
Total	146*	

* Point total for actual exam will be 100 points

I. Analysis – Class Design (12 points)

Assume that `class A` has been defined and implemented in Java, where `class A` defines public and protected functions as well as protected and private data.

`class A` encapsulates a large, stable data set and provides access via public functions. The calculation of summary statistics can be expensive and not necessarily informative since the data set does not change often.

Assume that two additional Java classes have been defined and implemented, each relating to `class A` in a different manner, as indicated below.

Evaluate these two Java classes `B` and `D`:

- 1) Define the functionality for one constructor for each class.
- 2) Identify what functions defined in `class A` are available to
 - a. Application programmers.
 - b. The class (designer of types) `B` and `D`.
- 3) Identify the implications of each design for future software maintenance.

```
1) Define constructor
// alternative #1
class B extends A
{
    // constructor

}
```

- 2) Functions and data `class B` available
 - a. to application programmer
 - b. to class designer

- 3) Implications of design for future software maintenance

```
1) Define constructor
// alternative #2
class D
{
    // constructor

    private A mine;
}
```

- 3) Functions and data `class D` available
 - a. to application programmer
 - b. to class designer

- 3) Implications of design for future software maintenance

Ila. Binding – C++ (18 points)

Answer the following questions given the following code:

```
class Parent
{
    int old;
public:
    Parent() { old = 0; }
    Parent(int old) { this->old = old; }
    void myFunction();
};
class Child: public Parent
{
    float data;
public:
    Child() { data = 0; }
    Child(int x): Parent(x) { data = 0; }
    Child(int x, float y): Parent(x), data(y) { }
    void myFunction() { cout << "Child myFunction\n"; }
};
class Child2: public Parent
{
public:
    Child2();
    Child2(int x): Parent(x) { }
    void myFunction() { cout << "Child2 myFunction\n"; }
};
class GChild: public Child
{
public:
    GChild();
    // TODO: constructors

    void myFunction() { cout << "GChild myFunction\n"; }
};
```

- a) (1 point) What binding is done in C++ by default?
- b) (2 points) Can you create an instance of a `Parent` object by calling one of the constructors defined in `Parent`? Why or why not?
- c) (1 point) What needs to be added in the code above to allow for dynamic binding?

d) (4 points) Complete the code for `// TODO: constructors` in the `GChild` class

e) (10 points) In `main`, write the code to demonstrate dynamic binding. All types in the `Parent` class hierarchy should be represented in a collection (call all available constructors). Once the collection is populated, call `myFunction()` on all `Parent` types.

```
int main()
{
```

```
}
```

IIb. Binding – Java (16 points)

- a) (10 points) How would you rewrite the code from IIa in Java?
- b) (2 points) Explain any differences between the languages with respect to dynamic binding.
- c) (2 points) Explain why a designer should judiciously choose dynamic binding.
- d) (2 points) Illustrate why a statically bound method in the parent class may negatively impact software maintainability.

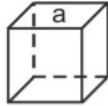



III. Shape3D (20 points)

Using a class diagram or in Java,

- 1) Create an interface and/or abstract class for 3D-Shapes, including functions:
 - a. `Volume()`
 - b. `SurfaceArea()`
- 2) Create data types Cube, Cylinder, Sphere, and Cone.
 - a. Define the function headers for core functionality, with accessibility noted.
 - b. Define fields essential to defining the Single Responsibility Principle.
 - c. Do not provide implementation details.
- 3) How may you refactor your 3D-Shapes design to promote reuse? (no code necessary)

In Java,

- 4) Write application code necessary to instantiate a heterogeneous collection of 3D-Shapes.

Name	Figure	Total surface area	Volume
Cube		$6a^2$	a^3
Right circular cylinder		$2\pi r(r+h)$	$\pi r^2 h$
Sphere		$4\pi r^2$	$\left(\frac{4}{3}\right)\pi r^3$
Right circular cone		$\pi r(l+r)$	$\frac{1}{3}\pi r^2 h$

IV. Copying (10 points)

- 1) Modify the C++ code below in the `ChangeMe()` and `Swap()` functions to pass by reference.

```
class SwapAndChange
{
    void changeMe(string s)
    {
        s = "change me";
    }
    public static void swapMe(int x, int y)
    {
        int temp = x;
        x = y;
        y = temp;
    }
}

int main()
{
    int a = 5, b = 10;
    string s;
    swapMe(a, b);
    changeMe(s);
    cout << "a: " << a << " b: " << b << " s: " << s << endl;
}
```

- 2) What will the output be after changing the functions?

In C++:

- 3) Which type, shallow or deep copying, is done by default?
- 4) Explain when you should consider deep copy. What functions need to be implemented for deep copy?

V. Comparative Definition or Evaluation – OO relationships (10 points)**

**Correctly fill in 10 slots for full credit; fill in all 12 for 1 point extra credit

Relationship	Lifetime	Association	Cardinality	Ownership
Composition ("has-a")				
Containment ("holds-a")				
Inheritance ("is-a")				

VI. Principles (10 points)

Consider the following design principle:

OPEN CLOSED PRINCIPLE (OCP)

Classes should be open for extension and closed for modification.

PRINCIPLE OF LEAST KNOWLEDGE

Every object should assume the minimum possible about the structure and properties of other objects

a) What does each principle say about the is-a, has-a, and holds-a relationships.

b) Identify the impact, if any, of each principle on the application programmer

c) How, if at all, do these two principles conflict?

XIII. Construction (40 points)

A poker **Game** may have 2-14 players. Poker games vary in the number cards dealt, the number of shared or “community” cards, and the number of cards that remain hidden.[†]

For the purposes of this exercise, each **Player** in a poker game will be dealt cards to make a complete **Hand**; a complete hand is defined as 5 cards. Assume that poker games may be played with more than one **Deck** of cards; a deck of cards consists of 26 cards.

The four main families or categories of poker games include:

- **Straight** – A complete hand is dealt to each player, face-down.
For simplicity, assume:
 - 5 cards face-down
 - 1 betting round
- **Stud** – Cards are dealt in a prearranged combination of face-down and face-up rounds, with a round of betting following each round.
For simplicity, assume:
 - 2 decks
 - 2 cards face-down, 3 cards face-up
 - 4 betting rounds
- **Draw** – A complete hand is dealt to each player, face-down, and after betting, players are allowed to attempt to change their cards by discarding unwanted cards and being dealt new ones.
For simplicity, assume:
 - 5 cards face-down
 - Optionally replace 1-4 cards
 - 2 betting rounds
- **Community** – Variation of stud poker. Players are dealt an incomplete hand of face-down cards, and then a number of face-up community cards are dealt to the center of the table, each of which can be used by one or more of the players to make a 5-card hand.
For simplicity, assume:
 - 2 decks
 - 2 cards face-down
 - 5 (community) cards face-up; players can choose 3 of these to make a 5-card hand.

A poker game has the ability to track the winner and has the following functionality:

- `Play()` – start playing a game
- `End()` – end game; once someone has won
- `Shuffle()` – shuffle deck(s) of cards before the start of new game

[†] <http://en.wikipedia.org/wiki/Poker>

Each game consists of players, who can do the following (as defined by functions):

- `Join()` – join a game; can only be a play 1 game at a time
- `Leave()` – leave a game
- `Check()` – pass; no bet
- `Bet()` – making the first bet; adding money to the pot
- `Raise()` – adding money to the pot, after the first bet
- `Call()` – matching a bet; necessary to stay in the game
- `Fold()` – discard one's hand and forfeit the money in the pot, including any money previously bet; usually happens when player does not want to `Call()` or match a bet

Assume that there is a poker **Casino** that offers the poker games described above. A casino is defined by games and players and has the following functionality:

- `Open()` – Players may enter the casino; games may be played
- `Close()` – All activities within the casino stop

Employing the OO tenets of abstraction, encapsulation and information hiding, and declaring all **relevant** functionality, use class diagrams or in Java,

- 1) Outline all the classes required for a fully functioning poker casino.
 - a. Ten classes have been defined (*italicized* and **bold**)
 - b. You may choose to add additional classes or interfaces
- 2) Define all relationships between classes through class definitions and defined fields.
- 3) Define the function headers for core functionality, with accessibility noted.

Include:

 - a. Constructors, where necessary
 - b. Public functions and properties, including parameters
 - c. Private fields that are essential to defining the Single Responsibility of the class
 - d. Self-documenting code
 - e. Concise comments, where necessary

Do not include:

 - a. Implementation details
 - b. Comments for given functions
- 4) How would an application programmer interface with the casino? What classes, and in turn, what functionality should be public?
 - a. Reflect these decisions in your design.
 - b. Put a star (*) next to the classes that may be instantiated by the AP.

Notes:

- Many elements have been simplified, do not complicate the design or add fields, properties, or functions that are not core to the definition of a particular class.
- It is completely valid to use some combination of class diagrams and Java code. However, please make it clear, and comment if there is any ambiguity or assumptions made.