Sample questions

Warning: almost no effort was made to “prettify” this document.

Answers are in white.

# Theory

1. What is O in SOLID, and describe its purpose. (actual past exam question)

2pt: O is the open and closed principal. 3pt: It says to extend rather than modify existing code.

1. OOP abuse is a code smell. Explain why. (actual past exam question)

It shows a lack of understanding of the problem, and can easily cause repeat code in the future

1. Name a disadvantage of using OOP.

Speed, possibility of improper use, etc

1. One of the ways we group languages is by generation. What is a generation (1-2 sentences)? Name a particular generation, and a language from that generation. (actual past exam question)

3 pt for what (age of the language based on the level it operates on), 2pt for generation (machine, etc) 2pt for matching language

1. The grammar or context-free portion of a language is called what?

**Syntax**

1. The meaning of a language is called what:

**semantics**

1. (NOT this time, but there will be a small task on this topic later) What does this EBNF produce. Provide an English explanation and two examples:

<S> -> a<S>c | b

This production rule generates the following set of strings:

{ b, abc, aabcc, aaabccc, … }

=> anbcn for n=0,1,2,3…

1. [22 point] (NOT this time, but there will be a small task on this topic later) Write an EBNF rule to produce American money. Specifically, it must have the following: (actual exam question, that I added additional test cases for the sample exam…which makes this too long for an actual exam)

* Have a leading dollar sign ($).
* A string of decimal digits.
* An optional fractional part with a decimal point (.) and two decimal digits.
* The string of digits to the left of the decimal point may consist of a single zero (0). Otherwise, it must not start with a zero.

Do not worry about commas. The equation editor has *ε*

The rule could produce:

* $1
* $100234056789
* $0.12
* $6.10
* $129.06
* $129.00

And the following should fail:

* $
* 1
* $1.
* $1.1.1
* $.12
* $.2
* $1.123
* $00.12

Possible answer:

*nzdigit* --> 1 | 2 | 3 | 4 | 5 | 4 | 7 | 8 | 9

*digit* --> 0 | *nzdigit*

*number* --> $ ( 0 | *nzdigit* { *digit }* ) ( ϵ | . *digit digit* )

When grading I checked again the following:

-2pt for **not** requiring $ sign (e.g. 1)  
-2pt for **not** allowing a 1+ string of numbers (e.g. $12345)  
-2pt for **not** allowing a 0 later string, potentially in the middle (e.g. $120345)  
-2pt for **not** allowing a no cents (e.g. $1)  
-2pt for **not** allowing 0.## (e.g. $0.12)  
-2pt for **not** forbidding 2+ preliminary 0 (e.g. 00.12 or 01.00 )  
-2pt for **not** forbidding . by itself , .#, .###, and/or .## (e.g. $1.1 or $.22 or . or $1.111111  
-2pt for **not** forbidding just $   
-2pt for **not** forbidding ###. (e.g. $1.)  
-2pt for **not** forbidding 2+ . (e.g. $1.1.1)  
-2pt for **not** using EBNF notation

# OOP

1. **Write a class called A, and derived B from it. Add functions to make A print “Hi” and B print “Bye.” B should override A’s version. You may do this in Java, C++, or pseudocode.**

**Class A**

**Print()**

**Print “hi”**

**Class B**

**Print()**

**Print “bye”**

1. Draw class diagram of the generic structure of the manager pattern (I can’t give you a drawing area on D2L, so this would be on paper)

This should be akin to an intermediate class connecting to another class with a \* multiplicity

1. Make a hierarchy tree such that all the following classes have one common ancestor. Specifically, state what inherits from what. Use parent <-- child notation. (actual past exam question)

* Organism
* Plant
* Mammal
* Feline
* Domestic cat
* Tiger
* Pine

1 pt each

Organism🡨mammal

Mammal🡨feline

Feline🡨cat

Feline🡨tiger

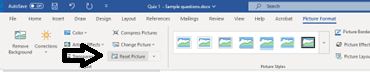
Organism🡨plant

plant🡨pine

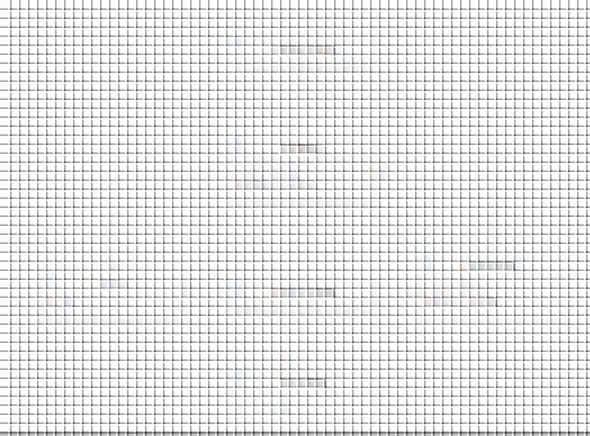
1. **(16 points)** Make a diagram of the following. Aim for a fairly pure OOP design. (actual past exam question)

*You are making a program that manages locks in an office building. Each building has employees, and the building has a name and address. Each lock has an id, and a schedule for each employee. Each employee has a schedule for each lock. An employee has a name and picture. A schedule has a list of hours that particular lock is open for that employee, in 24 hour time.*

**NOTE: To see the solution, select the image, go to the “Picture Format” tab, and choose “Reset”**

****

Here is an example solution:



**A couple of notes.**

* **An ID could just as well be a string. And picture could be a (URI) string**
* **Building could be your “hub” class.**
* **Since Schedule and Lock schedule are a 1 to 1 relationship, it is rational to put the hours in the LockSchedule.**
* **It is also rational to have the times in its own class. In this case, we know we have 24 of them, and the index is an exact map to the time, so it is OK to not have the manager class.**
* **What I don’t want to see is a \* to \* relationship between classes, which could happen if you tried to put the hours in Lock or Employee.**

1. Selection all of the following should use a manager. Your other options would be normal classes, struct-like structures, and inheritance. (actual past exam question)

Question options:

|  |  |
| --- | --- |
|  | In graphics, one item often owns another. Specifically, say you have a post that owns a propeller. Check if the propeller should use a manager. |
|  | You have a line with 2 distinct points that need to be called directly. |
|  | You have a collection of letters (the count is limited only by hardware) that will fill 0 or more boxes every 10 minutes. |
|  | You have an array of yards of at most 200 yards, that you need to make a schedule to mow them all. |

At minimum, a manager needs a collection of multiple type of sufficient size. This is so the implementation of the storage can change. The most restrictive is that a manager needs a collection of sufficient size, of one type, and there is an operation on the collection.

There are only 2 points in the line, and no operation. This is not a collection of sufficient size. Moreover, we could not call the points directly with a manager. A manager has no logical reason to be here.

You have a collection of letters, AND you have an operation that needs to look at sets of them. This meets the most restrictive requirements for a manager. It is an extremely good time to have a manager.

The post to propeller is a 1 to 1 relationship. No collection, no manager.

The yards may have a set max, but it is a collection of sufficient size. Also, there is a operation that needs to be done. This meets the most restrictive requirements for a manager. It is an extremely good time to have a manager.

1. What is the visitor’s patterns purpose?

It purpose it to provide a clean interface when you have an operation on diverse classes. Without this, maintaining single responsibility would be difficult as would the open/closed principle.

1. You have a video play list application with multiple media types already set up to accepts a Vistor that implements an interface called Visit. Visit does not have any default method implemented. The media types currently supported are MP4, GIF, and AVI. Create a new Visitor that will count the number of MP4 videos in the collection. You may do this in Java, Python, or Psuedocode. You may call the functions what you want as long as it is clear what its purpose is. (this problem was pulled from the exam this year for length!)

Solution: About 2 pt per line.

CountMP4 implements Visit  
 count = 0  
 acceptMP4( v : MP4)  
 count++  
 acceptGIF(v : GIF)  
 //empty  
 acceptAVI(v : AVI)  
 //empty

# Java

1. Write a for loop in Java that output even number from 0 to 10, inclusive

**No solution provided as this is best learning by coding it yourself in IntelliJ**

1. How do you write a constant in java?

**final static datatype name = x;**

1. Write a Java class called Point. (actual past exam question)

* Point must have private int attributes named x and y.
* Point must have a public constructor that takes in values for x and y.
* If no values are given in the constructor, set x and y both to 0.
* Point must have a public function named print() that outputs (x, y) using the values stored in x and y. For example, if x= 2, and y = 3, output:

(2, 3)

//example solution

//~2pt per line, 1 for syntax, and 1 for general purpose

//I permitted any access level for constructor, function, and class

class Point{

private int x = 0;

private int y = 0;

//this is still needed since java doesn't have default parameters

public Point(){ }

public Point(int x, int y){

this.x = x;

this.y = y;

}

public void print(){

System.out.println("("+x+", "+y+")");

}

}

1. In Java, complete the file named HelloWolrd.java that reads in a int and double from the console and store them in variables y and z respectively. You may skip the prompt and error checking. This is only ~6 lines of code. (actual past exam question)

**public class HelloWorld{**

**public static void main(String []args){**

**}**

**}**

//example solution

//~2pt per line other than the class and main (they were given)

// for each line, 1pt for syntax, and 1pt for general purpose

import java.util.Scanner;

public class HelloWorld {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int y = in.nextInt();

double z = in.nextDouble();

}

}