CSC 210 Fall 2024 – Mock Final Exam

This is an individual exam, and there will be a 120 minute window for you to complete it.

Do not cheat off of those nearby you.

Make sure to write your responses clearly and legible. There's no need to add comments or documentation to your code.

Each question has a designated box where your response should go. Use a DARK pen or pencil, and write INSIDE the answer boxes provided. Anything outside the box will **not** be considered as part of your response.

You may do extra work to arrive at the response, but the response MUST go **in** its designated box. Anything outside the box will **not** be considered as part of your response.

If you have any questions, please raise your hand. (regrade requests for "I didn't understand what the question was asking" are not going to be considered)

Question 1 – Read Code

Given this class:

```java
public class Foo {
    private String[] myArray = new String[16];

    public Foo() {
        for (int i = 0; i < 10; i++) {
            myArray[i] = "" + i;
        }

        for (int i = 10; i < 16; i++) {
            myArray[i] = "" + (char) (55 + i); // (char) 65 returns 'A'
        }
    }

    public String getIndex(int i) {
        return myArray[i];
    }

    public String[] getArray() {
        return myArray;
    }

}
```

What will the main method below print?

```java
public static void main(String[] args) {
    Foo myFoo = new Foo();

    for (int i = 0; i < 16; i++) {
      System.out.println(myFoo.getIndex(i));
    }
}
```

RESPONSE:

```
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
```

Question 2 – Enumeration (Exhaustive Search)

What will the code below print? Note that the code below makes use of the Foo class from the previous question.

```java
public class Enumeration {
    public static void main(String[] args) {
        Foo myFoo = new Foo();
        foo(2, "#", myFoo.getArray());
    }

    public static void foo(int n, String myString, String[] myArray) {
        if (n == 0) System.out.println(myString);
        else {
            for (String s : myArray) {
                foo(n - 1, myString + s, myArray);
            }
        }
    }
}
```

RESPONSE (write the first 5 and last 5 lines that are printed):

```
#00
#01
#02
#03
#04
.
.
.
#FB
#FC
#FD
#FE
#FF
```

Question 3 - Inheritance

Given the following classes:

```java
public class Fish {
    protected String message;

    public Fish() {
        message = "fish";
    }

    public String toString() {
        return message;
    }
}
```

```java
public class Red extends Fish {

    public Red() {
        super();
        message = "red " + message;
    }
}
```

```java
public class Blue extends Fish {

    public Blue() {
        super();
        message = "blue " + message;
    }
}
```

```java
public class HowMany extends Fish {

    public HowMany(int n) {
        super();
        message = n + " " + message;
    }
}
```

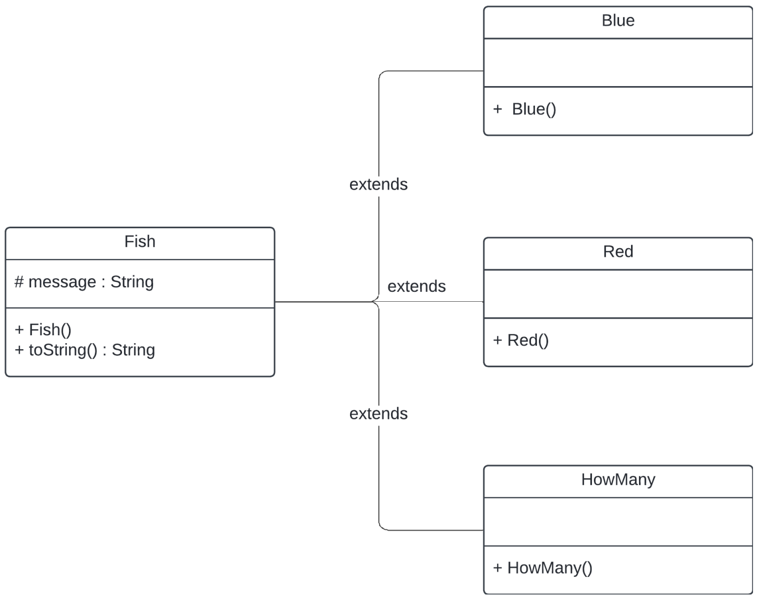
A) Which class is the super class?

Fish

B) What does the keyword `protected` mean?

The keyword ‘protected‘ in the instance variable for the super class means that the variable can be directly accessed by the subclasses

C) Draw the UML class diagram:



D) What will the code below print?

```
public class PrintFish {

    public static void main(String[] args) {
        Fish myFish = new Red();
        System.out.println(myFish);

        Fish myOtherFish = new Blue();
        System.out.println(myOtherFish);

        Fish anotherFish = new HowMany(3);
        System.out.println(anotherFish);
    }

}
```

red fish
blue fish
3 fish

Question 4

A) What does the keyword `this` represent in Java?

☐ reference to the super class
☐ a string representation of the current class
☐ a parameter to each constructor
☒ reference to the current object

B) What is the purpose of the `extends` keyword in Java?

☐ To define a new class as an interface
☒ To inherit methods and properties from another class
☐ To declare a method without implementation
☐ To implement an extended method

C) Which of the following statements about `extends` is NOT correct?

☒ A subclass can extend multiple classes using extends
☐ A class can extend another class
☐ A class can extend only one class
☐ A subclass inherits methods and properties from another class

D) What happens if you use `extends` to inherit a class in Java?

☐ All private members are inherited
☒ Only public and protected members are inherited
☐ Only static members are inherited
☐ No members are inherited; you must redefine them

E) What must a class do when it implements an interface?

☒ It must implement all methods of the interface
☐ It must be declared as abstract
☐ It must have the same name as the interface
☐ It automatically inherits all interface methods without overriding

F) What does the `abstract` keyword signify in Java?

☐ A class that cannot be extended
☒ A method or class that cannot be instantiated directly
☐ A method with an implementation in a subclass
☐ A class that must have a main method

G) Which of the following statements is true about an abstract class?

☐ It can have abstract methods but no concrete methods
☐ It must be declared final
☒ It can have both abstract and concrete methods
☐ It cannot be inherited

H) What happens if a class contains at least one abstract method?

☒ The class must be declared abstract
☐ The class can be instantiated
☐ The class must implement an interface
☐ The class cannot have any concrete methods

I) How do Java and Python handle variable types?

☐ Both Java and Python use static typing
☐ Java uses dynamic typing, and Python uses static typing
☒ Java uses static typing, and Python uses dynamic typing
☐ Both Java and Python use dynamic typing

J) Which of the following statements about Java and Python is correct?

☐ Java is interpreted, and Python is compiled
☒ Python is slower in execution compared to Java due to its interpreted nature
☐ Java is dynamically typed, and Python is statically typed
☐ Python does not support object-oriented programming like Java

K) Which of the following best describes memory management in Java and Python?

☐ Both Java and Python require manual memory management
☐ Java uses explicit garbage collection, whereas Python does not
☒ Both Java and Python rely on garbage collection for memory management
☐ Python uses manual memory management, while Java uses garbage collection

L) What is inheritance in Java?

☒ A mechanism where a class can acquire properties and methods of another class
☐ A way to implement multiple threads in Java
☐ A technique to hide the details of a class
☐ A method of defining interfaces in a class

M) What is the relationship between a parent class and a child class in Java?

☐ A parent class inherits from a child class
☒ A child class inherits from a parent class
☐ Both parent and child classes are independent of each other
☐ A parent class must be abstract if a child class is inheriting from it

N) What happens if a child class overrides a method of its parent class?

☐ The parent class method will always be called
☒ The child class method will override the parent method for the child object
☐ It is not allowed to override methods in Java
☐ The parent method becomes private to the parent class

Question 5 – Write Code

Write a method that takes in an integer `n` as argument and returns a string. The string should be `"FizzBuzz"` if `n` is divisible by 3 and 5, `"Fizz"` if `n` is divisible by 3, and `"Buzz"` if `n` is divisible by 5 or `n` (as a string) if none of the conditions are true.

RESPONSE:

```
public String fizzBuzz(int n) {
        String result = "";

        if (n % 3 == 0) result += "Fizz";
        if (n % 5 == 0) result += "Buzz";

        if (result.equals("")) result += n;

        return result;
    }
```

Question 6 – JUnit Tests

Write JUnit tests to test your method from the previous question.

RESPONSE:

```
@Test
public void testOne {
  assertEquals("FizzBuzz", fizzBuzz(15));
  assertEquals("Buzz", fizzBuzz(20));
  assertEquals("Fizz", fizzBuzz(21));
  assertEquals("4", fizzBuzz(4));
}
```

Question 7 – enum

Write a Java program that utilizes an enum for semester. The `enum` Season should have the following constants: SPRING, SUMMER, FALL, and WINTER

When the `Semester` class is instantiated, it should have a season plus the year. You should have a method to validate the year. Use encapsulation principles.

RESPONSE:

```java
public class Semester {

    public enum Season {SPRING, SUMMER, FALL, WINTER};
    private Season season;
    private int year;

    public Semester(Season season, int year) {
        this.season = season;
        if (validYear(year)) this.year = year;
    }

    public boolean validYear(int year) {
        String myString = "" + year;
        if (myString.length() != 4) return false;
        else return true;
    }

    public int getYear() {
        return year;
    }

    public Season getSeason() {
        return season;
    }

    public String toString() {
        String message = season + " ";
        message += year;
        return message;
    }
}
```

Question 8 – refactoring

The code below is attempting to manage a class roster (with multiple students). Each student has an unique ID, a name, and a major.

What problems do you see with this code? How would you fix them?

Consider both code smells and encapsulation.

```java
public class Roster {
    public HashMap<String, String> students;
    public HashMap<String, String> majors;
    public int count;

    public Roster() {
        students = new HashMap<>();
        majors = new HashMap<>();
    }

    public boolean addStudent(String id, String name, String major) {
        if (!students.containsKey(id)) {
            students.put(id, name);
            majors.put(id, major);
            count += 1;
            return true;
        }
        return false;
    }

    public int getCount() {
        int HowMany = 0;
        for (String s : students.keySet()) {
            HowMany += 1;
        }
        return HowMany;
    }
}
```

RESPONSE:

Encapsulation: instance variables should be private (they are all public), and there should be getters and setters for each instance variable

Unnecessary code: the getter `getCount()` uses unnecessary code to calculate how many students there are in the roster – there is an instance variable for that

Primitive Obsession: instead of two HashMaps, a separate class for `Student` should be written with private instance variables, getters and setters for: strings for name (possibly first and last separate), string for id, and enum for major. There should be a validation method for major. Roster should then have a collection of `Student` objects.

Question 9 - Generic classes

Write a generic class that takes four values of any type. Write setters and getters for it (follow the principles of encapsulation).

Here's a JUnit test, make sure your solution would pass this test.

```
@Test
void test() {
  MyClass<String, String, Float, Character> myObject = new MyClass();
  myObject.setA("0001");
  myObject.setB("Joanna");
  myObject.setC(98.5f);
  myObject.setD('A');
  assertTrue(myObject.getA() instanceof String);
  assertTrue(myObject.getB() instanceof String);
  assertEquals(myObject.getD(), 'A');
  assertEquals(myObject.getC(), 98.5f)
}
```

RESPONSE:

```
public class MyClass<A,B,C,D> {
    private A a;
    private B b;
    private C c;
    private D d;

    public MyClass(A a, B b, C c, D d) {
        this.a = a;
        this.b = b;
        this.c = c;
        this.d = d;
    }

    public A getA() {
        return a;
    }

    public B getB() {
        return b;
    }

    public C getC() {
        return c;
    }

    public D getD() {
        return d;
    }

    public void setA(A a) {
        this.a = a;
    }
    public void setB(B b) {
        this.b = b;
    }

    public void setC(C c) {
        this.c = c;
    }

    public void setD(D d) {
        this.d = d;
    }

}
```