This exam totals to 100 points. Follow the instructions. Good luck!

Chapter 5
This chapter was mostly dealt with objects expect questions similar to these.

1. Create accessors and mutators for the following class. Make sure to check for valid values

```java
public class Person {
    private String name;
    private int age;
    //Put accessors and mutators here
}
```
2. Write a method that converts and returns the instance variable “feet” to a
measurement in meters. The conversion between feet and meters is 1ft =
0.3048m

public class Measurement
{
    private double feet;
    //put the method here

}

3. Write a method that adds all the elements of an instance variable array together
and then prints the result. The method should only print and not return.
public class ArrayOfStuff
{
    private int[] array = {1,2,3,4,5};
    //Put the method here

}
Chapter 6
This chapter further expanded the concept of objects by introducing Constructors, Static methods, and Overloading methods.

1. Write a default constructor and a constructor that takes in parameters to set all the instance variables for the following class. Make sure to check for correct values.

```java
public class TaterChips {
    private int numberOfChips;
    private String brandName;
    private double netWT;
    //Write your constructors here
}
```

2. Write a static method that takes in two integers and returns the sum. Then implement that in the main method provided.

```java
public class MathSum {
    //Put your static method here
}
```

```java
public static void main(String[] args) {
    int number1 = 30;
    int number2 = 12;
    //Implement your method after the equals sign right here
    int number3 =
}
```
3. Create two overloaded methods that set the value of a password instance variable. One method should take in a string and the other should take in an integer.
   Hint(Integer.parseInt())

```
public class BriefcaseSecuritySoftware
{
    // Assumes passwords can only be numbers
    int password;
    // Put your methods here
}

```

Chapter 8
This chapter was about inheritance, polymorphism, interfaces and abstract classes. I won’t be testing anything on abstract classes.

1. We have a class Person that has the instance variable name and age, and has a constructor that takes both of those values in as parameters. Fill in the rest of the class Employee with the proper constructors: A default constructor that sets the employee number to a default value and also calls the Person’s default constructor, and another constructor that takes in the employee number along with the name and age. Make sure they call the parent’s constructor.

```
public class Employee 
{  
    int employeeNumber;  
    // Put your constructors here
}
```

2. We have a class **Animal** that contains the method public void printInfo(). Fill in the rest of the class **Gopher** by overriding the parent’s printInfo() method. It should call the **Animal**’s printInfo() method along with also printing the gopher’s address.

   ```java
   public class Gopher _________ //don’t forget to put something here
   {
       String holeAddress;
       //Put the overridden method here
   }
   ```

3. Write an interface called **Camera** with the following methods getBrand, setBrand, takePicture. The methods getBrand and setBrand return and set the brand (Assumed to be a string).

4. What is the difference between overloading and overriding methods?
Chapter 9
This chapter was on exceptions. You may expect to have questions creating exceptions, writing methods that throw exceptions, or writing code that uses methods that throw exceptions that require a try and catch block.

1. Create an exception class called **FatalCheeseException** that has two constructors: the default that calls the parent’s constructor with an error message, and another one that takes in a message string that is passed to the parent’s constructor.

```java
public class FatalCheeseException {
    // Don't forget to put stuff here
}
```

2. Write a method `eatCheese` in the given class `Cheese`. This method returns no values and takes in two parameters corresponding to the current month and year. Furthermore, this method could raise an exception if the current year or month is greater than the expiration date.

```java
public class Cheese {
    private String name;
    private int expMonth;
    private int expYear;

    // Write Method here
}
```
3. We have a class **Cheese** that has three instance variables name, expMonth, expDate. Along with those variables it has all of the accessors and mutators with error checking. This class also has a method **eatCheese** that can throw a **FatalCheeseException**. The method takes in a month and a year that corresponds to the current month and year. In the provided main method create a new instance of cheese setting its name, expMonth, and expYear. Also call its eatCheese method. Make sure if an exception occurs make sure to print the exception’s message.

```java
public static void main(String[] args)
{
    Scanner keyboard = new Scanner(System.in);
    System.out.println("I'm going to eat this cheese, but let me check if it's good. Enter the current month and year");
    int month = keyboard.nextInt();
    int year = keyboard.nextInt();
    //Put your code here calling eatCheese and handling the possible exception
}
```
Big Long Question
There will be one question that will require you to write a class from start to finish. Creating all constructors, accessors, mutators, and methods specified. It will be similar to what you have done in lab.

1. Create a class **Light** that has the following
   a. Two instance variables
      i. `isOn` – true or false the light is on
      ii. `bulbWattage` – an integer value corresponding to the number of watts in the bulb
   b. Two constructors
      i. Default – set the instance variables to a default value
      ii. One that takes in two parameters that will set the instance variables
   c. Accessors and Mutators for both variables
   d. A method `turnOnLight` that sets the value `isOn` to true
   e. A method `turnOffLight` that sets the value of `isOn` to false
This chapter focused on file input and output (File I/O). Understand how to read and write to and from a file.

1. In the method provided below, write code that will read a text file and count the number of times the word, “deer” appears. This should ignore any type of punctuation that may come at the beginning end of the word. Also make sure to print out any error message to the console if one were to occur.

   public static int deerCounter(String fileName)
   {
       //Enter your code here
2. In the method provided below, write code that will write to a given 2D matrix of numbers. You may assume that a normal space separates the numbers from left to right and a new line appears for every row. Make sure to print out a message to the console if an exception occurs.

```java
public static void matrixWriter(int[][] matrix, String fileName)
{
    //Put your code here
}
```