CS 143 Computer Science II (Java)

Midterm Study Guide

Midterm will be a "paper and pencil", "closed book" test. You will be not allowed to use books. You can bring only notes that fit on ONE 3x5 inch index card, written on both sides.

Midterm will cover topics from the following chapters of the text book

- Chapter 6 First Look at Classes
- Chapter 8 A Second Look at Classes and Objects
- Chapter 10 Inheritance
- Chapter 11 Exceptions and File I/O

The test will consist of **three parts**: multiple choice, short answer, and problem solving (writing the code of the problem solution).

To get ready for the multiple choice part of the test please make sure to work on the review questions and exercises at the end of each chapter. Answering "Checkpoint" questions is a very good test-prep activity too.

In order to feel more comfortable when writing code on paper, please consider solving "Algorithm Workbench" problems at the end of each chapter of the text book using just paper and pencil. Following are sample questions you may expect to see in the problem-solving part of the test.

Sample Problems for the Problem-Solving Part of the Test

1. Make sure to practice writing basic class code that includes
    a. Aggregation and references to objects as fields
    b. no-argument and regular constructors (that include memory allocation for aggregated objects)
    c. copy constructor (including copy constructor calls!) that create deep copies of referenced objects
    d. clone() method. Make sure you know the advantage of clone() method over copy constructors in case of inheritance. Make sure you know how to write a clone() method for class with primitive fields and for a derived class
    e. Accessor and mutator methods
    f. STATIC methods (make sure to know the limitations here!)
    g. STATIC fields
    h. toString() and
    i. equals() methods that compare not only fields of primitive types but objects like arrays
    j. this keyword and its meaning

2. Read UML diagrams and implement design described with UML diagram
3. Be prepared to write code that illustrates inheritance concepts such as overridden methods.
4. Write code that creates class that inherits from other class as well as implements an interface.
5. Practice answering the following Inheritance questions

    a. If class Dog inherits from class Pet, what member variables and what methods class Dog will have? Make up an example that goes with this question.

    b. Imagine that the child class was derived from the parent class with the public base class access specification.

        i. What parts of the parent class (variables and functions) are accessible from the child class?

        ii. How the public and private modifiers change the accessibility?

    c. What are the advantages of using "protected" access specification within the class?

    d. What are the rules of writing a constructor of a derived class? Make up an example – write a non-default constructor for class Dog that is derived from class Pet.

    e. Can we redefine base class methods in derived class?

    f. Terminology: what is the difference between overriding and overloading of a method?

    g. What is an abstract class? Can we instantiate an abstract class (create objects of such type)? Why not?

    h. Practice writing an abstract class called MyClass that has field of type double myDouble and has an abstract method int myMethod(int k). Practice writing a class that inherits from the abstract class, call it ClassA. Implement the abstract method. Let's have this method find the sum of all integers up to k (1+2+3+…+k).

    i. What is interface?

    j. How is it different from a class? Can an interface have a regular field?

    k. What does it mean to implement an interface?

    l. Can a class implement more than one interface?

6. Practice answering questions on exceptions

    a. What are the benefits of exception mechanism? What kind of error-handling can be achieved only with exceptions, but not with if-else statements?

    b. How do we throw an exception? How do we catch an exception? Write several lines of code that demonstrate the concept: method throws exception, calling code catches and handles it.

c. What role does the type of exception play? Can we have more than one catch block? Make up an example where we have 2 or more catch blocks and more than one function calls in the try block that may throw exceptions.

d. What is an exception class? Why would we may want to create and use one?

e. What happens when exception is not caught?

f. Practice writing methods that throw exceptions.

g. Practice writing code that catches and handles exceptions. Make sure to roughly know the inheritance tree of exceptions and the rules associated with the order in which you write catch blocks.

7. Binary file I/O questions

   a. Make sure to know the syntax of opening a file for writing in binary format and writing into it

   b. Make sure to know the syntax of opening a file for reading in binary format and reading from a file

   c. What is serialization? Why would you make a class to implement a Serializable interface? What are the methods of Serializable interface?

   d. Practice writing code that opens a binary (or text) file, reads all numbers from the file, finds the largest/smallest without storing any numbers in array, closes the file.

   e. Make a method out of it. The method would be taking a file name as a parameter and return smallest number in file. The method will be throwing exception.

   f. Catch and handle exceptions in the calling code

   g. Practice writing code that creates a new file and writes some binary output into it. Catch and handle any exceptions that arise.

In addition, please make sure to practice solving problems appearing in Quiz 1 and 2 preparation documents, copied below.

From Quiz 1:

Sample Problems

Given a class described below:

```
public class Rectangle {

   private double length;

   private double width;

   public Rectangle(double len , double wid) {/*implemented*/}public
   Rectangle(){/*implemented*/}

   public Rectangle(Rectangle obj) {/*implemented*/}
```

```
   boolean equals(Rectangle) {/*implemented*/}
   public String toString(){/*implemented*/}
   /* accessor and mutator methods implemented*/
}
```

Write a class MyClass that has 2 fields: Rectangle r and int aField. Implement the following methods / fields to the new class:

1. Accessor for field r. Make a deep copy of the field when returning it.
2. Constructor MyClass(int a, Rectangle b) that sets both fields. Make a deep copy of object b when copying it!
3. No-argument constructor that sets aField to 0, and r field to a rectangle object with length and width set to 1.
4. Mutator for aField. The value stored in that field must not be negative. Throw Illegal ArgumentException when the negative value is detected.
5. copy constructor for MyClass
6. toString() method for MyClass
7. equals() method for MyClass
8. A static field to MyClass that counts how many objects of that class were created. Make adjustments to the constructors as needed to support the object counting feature of MyClass.
9. A static accessor method to return the static field

From Quiz 3:

Sample Problems

Given a class described below:

```
public class Rectangle implements Cloneable{
   private double length;
   private double width;
   public Rectangle(double len , double wid) {/*implemented*/}
   public Rectangle(){/*implemented*/}
   public Rectangle clone(){/*for you to implement*/}
   @Override boolean equals(Object obj) {/*for you to implement*/}
   public String toString(){/*implemented*/}
   /* assume accessor and mutator methods implemented*/
}
```

1. Implement clone() method for Rectangle class
2. Implement equals() method for Rectangle class
3. Write a definition of a class MyRectangle that inherits from Rectangle and has a field of type int[] myArray. Class must also implement Cloneable and Relatable interfaces.
4. Write constructor *public MyRectangle(int len, int wid, int[] array*) that sets all the fields of the class. Make sure to
    a. call the constructor of the base class first
    b. make a deep copy of an array given to you.
5. Implement clone() for MyRectangle. Make sure to do it according to the "black magic recipe" we discussed in class.
6. Implement **`@Override boolean equals(Object obj)`** for MyRectangle
7. Implement *toString()* method for MyRectangle class
8. Use Rectangel and MyRectangle classes to illustrate the concept of polymorphism in OOP.
9. Write a definition of MyAbstract class that has a field of type int and an abstract public method *int doStuff()*
10. Write a definition of a MidTermable interface that has the following methods: *boolean readTextBook(); boolean practiceCodeWriting();*
11. Add a field to the interface int N. Make sure the field is defined as a CONSTANT

Additional Practice Problems for Binary File I/O and Exceptions:

1. Write a segment of code that
    a. Creates an array of size 100 of type Rectangle, fills that array with Rectangles of random integer widths and lengths. The range of widths and lengths must be [1, 100].
    b. Then creates a binary file "data.dat" and writes all 100 rectangles into the file using a loop.
    c. Make sure to use Serializable interface in your solution.
    d. Make sure to use try/catch block when manipulating with files.
2. Make a new exception class NegativeMeasurementException with no custom messages.
3. Write a constructor `public Rectangle(double len , double wid)throws NegativeMeasurementException.` The exception is thrown when length or width are negative.