CISC 3120 Midterm Review Guide

Main Topics

Java Fundamentals

I expect you to have a basic mastery of control structures (loops, conditionals), statements, arrays, function declaration, definition, and invocation, and language syntax. I won't be testing you explicitly on this, but of course you won't be able to write much code without this! Exam questions will focus on:

Basic syntax, terminology (class, object, instance variable, subclass, etc)
Primitive vs reference types.
Java API, packages.
Public, private, and package access.
Basics of inheritance in Java: the IS-A relationship, subtype polymorphism
Overloading and overriding
Object lifecycle: garbage collection
Static versus instance members

Professional Practice

Software Design Principles and Style
Version control and Git/GitHub

Study Tactics

Review all book exercises to make sure you understand why correct answers are correct.
Review all code from lecture/lab.
Review CodeLab exercises.
Review Application Activities (below)
Re-read readings now that you've worked with ideas in lecture/lab.

You should be able to write a basic class/application in Java without consulting notes. This includes using `main()`, `System.in`/`System.out`. If I ask specific questions about (say) the `Scanner` class, I will provide you enough of the Java API for you to answer the question.

Exam Structure, Roughly

50%: short-answer/multiple-choice
30%: writing code
20%: analyzing code

Why is the Java Virtual Machine that runs Java programs called “virtual”?
a) It is run over a network, rather than on the same physical machine on which it appears the Java program is running.
b) It is a software machine, rather than a “real” hardware machine that can run (say) a compiled C++ program.
c) It is a conceptual machine that is useful during the compilation process, but “real” hardware actually runs the program.
d) It operates on bytecodes, rather than the low-level bits that are manipulated by hardware.

Which of these are advantages of the virtual machine approach?
a) Once a program is written, it can be run on any platform with a JVM.
b) More security checks can be made before a program runs.
c) More people can write Java programs.
d) Both a and b.
e) Both a and c.

Which of the following types is not a primitive type?
a) boolean
b) int
c) String
d) double
e) char

If a variable is not primitive, it is a
a) Pointer
b) Object
c) Reference
d) Constant
e) Class

A reference to an object is best thought of as
a) The address of the object in memory.
b) Another name for the object.
c) A way to get to the object in order to tell it to do something.
d) A way to restrict what the object can do.

How are parameters passed in Java?
a) By value
b) By reference
c) By pointers
d) Both a and b
e) All of a, b, and c

If an instance variable is declared private, which methods will be allowed to access it?
a) All methods belonging to the same object/instance
b) All methods that have the same return type as the instance variable
c) All methods belonging to objects of the same class
d) All methods belonging to objects in the same application

If this code will compile and run correctly, what is its output? If it will not compile/run, explain why. (Note: this is similar, but not identical, to the code on p. 63.)

```
class Book {
      String title;
      String author;
}

class BookTestDrive {

      public static void main(String [] args) {
            Books [] myBooks = new Book[3];
            myBooks[0] = new Book;
            myBooks[1] = new Book;
            myBooks[2] = new Book;

            myBooks[0].title = “The Grapes of Java”;
            myBooks[1].title = “The Java Gatsby”;
            myBooks[2].title = “The Java Cookbook”;

            myBooks[0].author = “bob”;
            myBooks[1].author = “sue”;
            myBooks[2].author = “ian”;

            Book bookref = myBooks[0];
            i = 0;
            while (i < 3) {
                  System.out.print(bookref.title + “ by ” + bookref.author);
                  i = i + 1;
            }
      }
}
```

What happens if you don’t initialize a variable?
a) It has an undefined value, which could cause problems when the program runs.
b) If this program needs an initial value when it runs, it will throw an exception and halt.
c) The compiler will issue an error, and you’ll have to add the initialization before the code will compile.
d) The value is automatically set to some form of zero.

12. Using the following declarations:

```
int i, j;
double d;
char c;
String s,t;
```

and assuming that we have some code that causes these variables to get non-zero values, which of these comparisons is illegal (will cause a compiler error)?

a) c == i
b) c == d
c) s == t
d) s.equals(t)
e) None of the above are illegal

13. Write a Java program that reads exactly 10 integers from the keyboard, stores them in an array, calculates the average, then outputs the list of numbers and their average.
Write a function called average() that returns the average of an array of integers.

14. If the output of this Java program (excluding prompts) is

```
 o  The mppn is made pf green cheese.
```

what was the input?

```
import java.util.Scanner;
public class FindInput {

      public static void main(String[] args) {
            int x = 0;
            String s = "";

            Scanner cin = new Scanner(System.in);
            System.out.print("Input, please: ");
            if (cin.hasNextInt()) {
                  x = cin.nextInt();
            }
            if (cin.hasNextLine()) {
                  s = cin.nextLine();
            }

            if (x % 2 == 0) {
                  Widget w = new Widget(x, s);
                  System.out.println(w.widge());
            }
            else {
                  Gidget g = new Gidget(x, s);
                  System.out.println(g.gidge());
            }
      }
}

public class Widget {
      private int i;
      private String s;
      public Widget (int i, String s) {
            this.i = 2*i;
            this.s = s;
      }

      public String widge() {
            char c = s.charAt(i);
            return (c + " " + s.replace(c,++c));
      }
}

public class Gidget {
      private int i;
      private String s;
      public Gidget(int i, String s) {
            this.i= 3*i;
            this.s = s;
      }
```

```
        public String gidge() {
                Widget w = new Widget(i,s);
                char c = s.charAt(i);
                s.replace(c,++c);
                return (s.charAt(i) + w.widge());
        }
}
```

What's weird about the method invocation Integer.parseInt("3")?
a) "3" is a String, not an int, you fool.
b) You invoke methods after the name of an instance, not the name of a class, you dunderhead.
c) "Parse" means "to examine closely," you flea-ridden son of a compiler.
d) In Java, integers are primitive values, not class instances, you arrant knave.
e) There's nothing weird about this at all, you unsavory rapscallion.

What's wrong with this fragment of code from the book (p. 152)?

```
private static final String alphabet = "abcdefg";
private int gridLength = 7;
private int gridSize = 49;
```

a) Shouldn't alphabet *not* be static?
b) Isn't alphabet missing, like, 19 letters?
c) Shouldn't gridLength and gridSize also be constants?
d) Shouldn't gridSize be initialized to gridLength*gridLength?
e) Both c and d.

What's wrong with this fragment of code from the book (p. 152)?

```
while (!success & attempts++ < 200 ) {
        location = (int) (Math.random() * gridSize);
        //System.out.print(" try " + location);
        int x = 0;
        success = true;
```

a) Logical "and" in Java is a double &&, not a single &.
b) What the heck is that 200 for?
c) The whole loop condition needs more parentheses to avoid order-of-operations problems.
d) The (int) cast is unnecessary.
e) In the argument to print(), you can't "plus" a String and an int.

The Java API has many packages, including java.util, java.util.concurrent, and java.util.concurrent.atomic. What is the relationship among these three packages?
a) The names suggest how the behaviors of the packages are related, but to the compiler/JVM they are just three different packages.
b) They are "nested" in the sense that when you write import java.util.*, you get the stuff in all three packages.
c) They are "nested" in the sense that you cannot write import java.util.concurrent.atomic.* unless you've already written import java.util.* or import java.util.concurrent.*.
d) They are "nested" in the sense that "higher" packages (like java.util) automatically have access to all the public- and package-access code in the "lower" packages (like java.util.concurrent).

Are these two loops equivalent (they leave sum and niftyArray in the same state)?

```
for (int i=0; i < niftyArray.length; i++) {
{
      sum += niftyArray[i];
}
```

```
for (int i : niftyArray)

      sum += i;
}
```

a) Yes.
b) No.

Are these two loops equivalent (they leave sum and niftyArray in the same state)?

```
for (int i=0; i < niftyArray.length; i++) {
      sum += niftyArray[i];
      niftyArray[i] += 10;
}
```

```
for (int i : niftyArray) {
      sum += i;
      i += 10;
}
```

a) Yes.
b) No.

The method heidiPurge() has the following signature:

```
void heidiPurge(ArrayList<int> yerIn, ArrayList<int> yerOut, ArrayList<int> target);
```

When the function returns, target should not contain any elements of yerOut, but should contain all elements of yerIn (except, of course, any that are also listed in yerOut). Implement this function, taking full advantage of the ArrayList API.

Consider the following code (from p. 169):

```
public class Doctor {
      boolean worksAtHospital;

      void treatPatient() {
            // perform a checkup
      }
}
```

```
public class FamilyDoctor extends Doctor
{
      boolean makesHouseCalls;

      void giveAdvice() {
            // give homespun advice
      }
}

public class Surgeon extends Doctor {
      void treatPatient() {
            // perform surgery
      }

      void makeIncision() {
            // make incision (yikes!)
      }
}
```

Which of these is true?

a) A FamilyDoctor can treatPatient().
b) A FamilyDoctor can makeIncision().
c) A Surgeon treatPatient() differently than a FamilyDoctor does.
d) Both a and c.
e) Both a and b.

"Override" means
a) A subclass automatically gets the base class implementation of a method, even if it's not stated explicitly.
b) A method in a subclass can substitute the base class implementation with its own.
c) The base class implementation of a method is used even if a subclass has its own implementation.
d) A class can provide multiple implementations of a method.
e) If the warp core is about to fail, switch to auxiliary power.

In general, overridden methods should provide
a) Behaviors unique to the subclass.
b) Behaviors common to all subclasses.
c) Behaviors that may change in future implementations.
d) Behaviors that add to the base class behavior.
e) Behaviors that the JVM cannot support.

Consider the following code:
```
class Hamburger {                      public class Cheeseburger extends Hamburger {
      private int calories = 800;            int slices;
      int getCalories() {                    Cheeseburger(int slices) {
            return calories;                    this.slices = slices;
      }                                      }

}                                            @Override
                                             int getCalories() {
                                                return super.getCalories() + 100 * slices;
                                             }
                                       }
```

What is the result of trying to compile and run the following code?

```
Hamburger[] breakfast = new Hamburger[4];
breakfast[0] = new Cheeseburger(1); //Hamburger();
breakfast[1] = new Cheeseburger(2); //Hamburger();
breakfast[2] = new Cheeseburger(1);
breakfast[3] = new Cheeseburger(2);

int calories=0;
for (Hamburger h : breakfast) {
      calories += h.getCalories();
}
System.out.println("Your total calories for the meal: " + calories);
```

a) A compilation error.
b) A runtime exception complaining about a type mismatch.
c) Your total calories for the meal: 3200
d) Your total calories for the meal: 3500

Given the method declaration
```
public int moshify(double goodstuff[], String name)
```

which of these is a legal overload?

a) public double moshify(double goodstuff[], String name)
b) private int moshify(double goodstuff[], String name, ArrayList<int> badstuff)
c) public int moshify (int goodstuff[], String name)

d) private int moshify(double badstuff[], String city)
e) Both b and c.

An abstract class cannot be
a) Subclassed
b) Changed
c) Instantiated
d) Overridden
e) Compiled