Computer Sciences 302
Final Exam Information & Sample Questions

Below you’ll find information about the final exam and sample exam questions. This sampling does not represent the length or difficulty of the actual exam. The exam will have more multiple-choice questions and cover a broader range of topics than what is presented here. See the course web pages for a list of exam topics as well as the exam time and rooms. Note the following logistics about the exam:

- The final exam will be composed of two parts, about 3/4 multiple-choice and about 1/4 written questions. For the multiple-choice part, we’ll provide a bonus question, but there will be no extra credit if you get all of the questions right.
- Note the multiple-choice questions will be cumulative and the written questions will focus on the material since the second exam (but they are still dependent on earlier programming concepts).
- The exam is designed to be 120 minutes in length. Use your time wisely. You’ll be required to turn in your exam after two hours.
- As with prior exams, we’ll provide descriptions of common methods for select classes from the Java API where needed. For this sample you should use the Java API documentation and your textbook as needed.
- You may not use textbooks, notes, neighbors, calculators, or other electronic devices on the exam.
- Bring several #2 pencils and your UW ID to the exam.
- Sit in columns with an empty seat or an aisle to your right and left.
- Before the exam begins, pick up a scantron form and write on it your UW ID number (be careful, this is required to upload your score into Learn@UW). Then drop off your UW ID at the front of the room and fill in your name and other information as described on the exam.
- When you’re done, turn in your exam and scantron form, and pick up your UW ID.

When the exam starts, check the written questions and make sure to allocate sufficient time to complete them. We recommend that you first work on the multiple-choice questions that you feel confident answering. Second, read through the written questions and spend some time working on them. Then, go back and answer the remaining multiple-choice questions. Finish by completing your work on the written questions. Be careful not to be stubborn or get stuck on a question. If you’re having difficulty, go on to the next question.
Part I  Sample Multiple-Choice Questions

Questions 1 - 8 show examples of multiple choice questions for material up to the second exam. Questions 9 - 13 are on the new material. For questions for material from the first exam, see that exam and the exam 1 materials that were provided.

1.) Assume a class named Movie has a method named getTitle that returns a Movie object's title. Also assume an array named database has been filled with Movie objects. Which one of the following correctly tests whether the title of the first movie in the database is equal to "Koyaanisqatsi"?

A. database[0].getTitle() == "Koyaanisqatsi"
B. database[1].getTitle() == "Koyaanisqatsi"
C. database[0].getTitle.equals("Koyaanisqatsi")
D. database[1].getTitle.equals("Koyaanisqatsi")
E. database[0].getTitle().equals("Koyaanisqatsi")

2.) Consider the following complete implementation of the Question class:

```java
public class Question {
    public final double A = 11.22;
    public double b;
    private double c;
    public void one (double d) {
        this.b = d;
    }
    public double two () {
        return this.c;
    }
    public void three (double e) {
        this.c = e;
    }
}
```

Assume q is a properly constructed Question object, and it and the statements below are in the main method of an Exam class. Which one of the following statements will not result in a compiler error?

A. q.A = 1.1;
B. q.one(q.c);
C. q.two(q.b);
D. double d = q.two(11);
E. q.three(q.b);

3.) Which one of the following statements about Java data types is false?

A. Data types are classified into two groups: primitives and references.
B. Java allows arrays of primitive data types and arrays of reference data types.
C. A primitive value is copied whenever it is passed as a parameter or returned from a method.
D. A software object is copied whenever its reference is passed as a parameter or returned from a method.
E. Reference variables can be aliases but primitive variables cannot.
4.) Consider the following memory diagram:

Which one of the following code fragments corresponds with this memory diagram?

A. Account a1, a2;
B. Account a1, a2;
   a1 = new Account();
C. Account a1, a2 = new Account();
D. Account a1 = new Account();
   Account a2 = new Account();
E. Account a1 = new Account();
   Account a2;
   a2 = a1;

5.) Consider the following complete implementation of the Data class:

```
public class Data {
    private int data;
    public int get () {
        return data;
    }
    public void set (int d) {
        data = d;
    }
}
```

Also consider the following application class:

```
public class DataApp {
    public static void main(String[] args) {
        int data = 11;
        Data d1 = new Data();
        d1.set(data);
        Data d2 = d1;
        method(d1);
        data = d2.get();
        d2 = new Data();
        d2.set(33);
        System.out.println(data + ', ' + d1.get() + ', ' + d2.get());
    }
    public static void method(Data d) {
        d.set(22);
    }
}
```

What is printed when the DataApp program is executed?

A. 11, 11, 33
B. 11, 33, 33
C. 22, 22, 33
D. 22, 33, 33
E. 33, 33, 33
The questions on this page are based on the following code and each question has only FOUR choices:

```java
public class SemesterRecord {
    private ArrayList<Course> courses;
    private int i;
    public SemesterRecord() {
        this.courses = new ArrayList<Course>();
    }
    public void addCourse(Course c) { //SEE QUESTION BELOW }
    public int getSize() {
        return this.courses.size();
    }
    public void rewind() {
        this.i = 0;
    }
    public boolean hasNext() {
        return this.i < this.courses.size();
    }
    public Course getNext() {
        Course c = this.courses.get(this.i);
        this.i++;
        return c;
    }
}
public class Course {
    private String name;
    private int credits;
    public Course(String n, int c) {
        this.name = n;
        this.credits = c;
    }
    public String getName() {
        return this.name;
    }
    public int getCredits() {
        return this.credits;
    }
}
```

6.) Assume that a SemesterRecord, named sm, has been properly constructed and 0 or more Course objects have been added to it. Consider the code fragment below that is intended to find the position of a course named “cs302” if it is in the semester record:

```java
int found = -1;
sm.rewind();
for (int i = 0; CONDITION1; i++) {
    if (CONDITION2) {
        found = i;
    }
}
```

Which one of the following replacements for CONDITION1 and CONDITION2 when used to complete the code fragment above will ensure that it works as described?

- A. sm.hasNext() sm.getName() == "cs302"
- B. sm.hasNext() sm.getNext().getName().equals("cs302")
- C. i < sm.getSize() sm.getNext().getName() == "cs302"
- D. i < sm.getSize() sm.getName().equals("cs302")

7.) Which one of the statements below, when added to the body of the addCourse method above correctly adds the course to beginning of a SemesterRecord object?

- A. courses.add(c);
- B. return this.courses.add(c);
- C. this.courses.add(0, c);
- D. return this.courses.add(0, c);

8.) Which one of the following method headers would result in overloading when its method is added to the SemesterRecord class above?

- A. public void addCourse(Course c, int index)
- B. public boolean rewind()
- C. public Course getCourse(int index)
- D. private Course getNext()
The questions on this page are based on the following partially shown code:

```java
class Exceptions {

    public static void main(String[] args){
        methodX();
        try {
            // LINE A - see questions below
        } catch (NullPointerException x) {
            System.out.print("Error 1,\n");
        } catch (IndexOutOfBoundsException x) {
            System.out.print("Error 2,\n");
        }
        System.out.print("main done,\n");
    }

    public static void methodX() {
        try {
            // LINE B - see questions below
            methodY();
        } catch (NullPointerException x) {
            System.out.print("Error 3,\n");
        }
        System.out.print("methodX done,\n");
    }

    public static void methodY() {
        try {
            // LINE C - see questions below
        } catch (ArithmeticException x) {
            System.out.print("Error 4,\n");
        }
        System.out.print("methodY done,\n");
    }

    //
}
```

9.) What is output if LINE A is replaced with code that will throw a NullPointerException:
   A. Error 1,main done,
   B. methodY done,methodX done,Error 1,
   C. methodY done,methodX done,Error 1,main done,
   D. methodY done,Error 3,methodX done,main done,
   E. The program crashes and message about the exception is displayed.

10.) What is output if LINE B is replaced with code that will throw an IndexOutOfBoundsException:
    A. Error 2,
    B. Error 2,main done,
    C. methodY done,Error 2,main done,
    D. methodY done,methodX done,Error 2,main done,
    E. The program crashes and message about the exception is displayed.

11.) What is output if LINE C is replaced with code that will throw a NullPointerException:
    A. Error 4,methodY done,methodX done,main done,
    B. methodY done,Error 3,methodX done,main done,
    C. Error 3,methodX done,Error 1,main done,
    D. Error 3,methodX done,main done,
    E. The program crashes and message about the exception is displayed.
12.) Which one of the following statements about Java exception handling is false?
   A. There are three categories of exceptions: checked, unchecked, and runtime exceptions.
   B. Unchecked exceptions are only detected during runtime.
   C. Checked exceptions are checked at compile time and require either a catch block or a throws clause.
   D. Checked exceptions include FileNotFoundException.
   E. Runtime exceptions would be thrown when dividing a number by 0 or trying to access a null object.

13.) Consider the following code fragment where fileIn is a properly constructed Scanner object and fileOut is a properly constructed PrintWriter object:

   ```java
   while (fileIn.hasNextLine()) {
     String s = fileIn.nextLine();
     if (s.charAt(0) == '#') { fileOut.println(s); }
   }
   fileIn.close();
   fileOut.close();
   ```

   If fileIn opens an input file containing the following:

   ```
   #line1
   line2#
   ##line3
   ```

   Which one would be the contents of the output file after the method has completed?

   A. line2#
   B. #line1##line3
   C. line1
      line1
      line3
   D. #line1
      ##line3
   E. line1
      line2#
      #line3
Part II Sample Written Questions

Here are samples of written questions for some of the new material topics. As with the second midterm exam, it is ok to use the following abbreviations \texttt{S.o.pln()} for \texttt{println} and \texttt{S.o.p()} for \texttt{print}, and you will \textit{not} need to comment your code. Unlike the second exam, you may \textit{not} assume that non-null references are passed to methods. If reference parameters are \texttt{null} throw an \texttt{IllegalArgumentException}.

1.) \textbf{Complete the method below} that is passed two strings, a name of a file and a target line. This method returns \texttt{true} if the file contains at least one line that matches the target line, and \texttt{false} otherwise. Note your method must include code to properly do the steps needed for file input, but you may assume that any necessary import statements have already been coded.

\begin{verbatim}
public boolean containsLine (String filename, String target) {
\end{verbatim}
2.) Consider the Timer class below:

```java
public class Timer {
    private int seconds; //legal values: 0 to 59
    private int minutes; //legal values: >= 0
    private int limit;   //legal values: >= 1, or 0 means no limit

    public Timer () {
        seconds = minutes = limit = 0;
    }

    public void reset() {
        seconds = minutes = 0;
    }

    public boolean tick() {
        seconds++;
        if (seconds >= 60) {
            seconds = 0;
            minutes++;
        }
        if (limit != 0 && minutes >= limit) {
            return true;
        }
        return false;
    }

    public void setLimit(int limit) {
        if (limit >= 0) {
            this.limit = limit;
        }
    }

    public int getSeconds() {
        return seconds;
    }

    public int getMinutes() {
        return minutes;
    }
}
```

A. Modify the code above so that it implements the Comparable interface. Show the changes and additions needed to the right of the code above. Note you do not need to rewrite the entire class, just show the modifications.

B. Assume two Timer objects have been created, named t1 and t2. Write a code fragment that displays the minutes and seconds of the timer with the lowest time or "Tie" if they are the same time.

C. Rewrite the tick method so that, rather than returning a boolean value, it throws an AlarmTriggeredException when the minutes value reaches the limit. You may assume the AlarmTriggeredException class has been coded and extends the Exception class.

D. Rewrite the code below so that it is functionally equivalent but uses the new version of tick as described in part C above.

```java
while (!t1.tick()) {
    System.out.println("tick");
}
t1.reset();
```