CS 1331 A/B Exam 2

Ruirui Ma

TOTAL POINTS

95 / 100

QUESTION 1

M/C 28 pts

1.1 hashcode and equals 3 / 3

✓ + **3 pts Correct**

+ **0 pts** Correct Answer was 4

1.2 interface 3 / 3

✓ + **3 pts Correct (2 or 3)**

+ **0 pts** Correct answer was 2 or 3

+ **0 pts** No answer

1.3 for-each 3 / 3

✓ + **3 pts Correct**

+ **0 pts** Correct answer was 3

+ **0 pts** No answer

1.4 sys-args 3 / 3

✓ + **3 pts Correct**

+ **0 pts** Correct answer was 3

1.5 array 3 / 3

✓ + **3 pts Correct**

+ **0 pts** Correct answers was 5

1.6 dynamic type 3 / 3

✓ + **3 pts Correct**

+ **0 pts** Correct answer was 5

1.7 extend 2 / 2

✓ + **2 pts Correct**

+ **0 pts** Correct answer was False

1.8 abstract class 2 / 2

✓ + **2 pts Correct**

+ **0 pts** Correct answer was False

1.9 static type - interface 2 / 2

✓ + **2 pts Correct**

+ **0 pts** Correct answer was True

1.10 unboxing 2 / 2

✓ + **2 pts Correct**

+ **0 pts** Correct answer was False

1.11 super 2 / 2

✓ + **2 pts Correct**

+ **0 pts** Correct answer was False

QUESTION 2

2 Equals 7 / 8

+ **1 pts** Did not reference equality check (this == other), but no deduction made.

✓ + **1 pts Reference equality check (this == other)**

+ **8 pts** Correct

✓ + **2 pts Properly overrides equals method from Object**

✓ + **1 pts instanceof check (!(other instanceof Car))**

✓ + **1 pts Properly casts Object to a Car**

✓ + **2 pts Checks model, horsepower, and turbo for equality**

+ **1 pts** Checks model, horsepower, turbo, and wheelsize

✓ + **1 pts Returns a boolean**

- **2 pts** Major syntax error

✓ - **1 pts Compares Strings with "=="**

- **1 pts** Uses getter methods instead of instance data

- **1 pts** Did not compare turbos

QUESTION 3

3 Enum 8 / 8

✓ + **8 pts Correct**

+ **4 pts** Correct enum header (enum Day)

+ **4 pts** Has values for each day of the week

+ **0 pts** Incorrect

- **2 pts** Major Syntax Error

- **2 pts** Enum name incorrect (eg. should be Day, not DAY)

- **2 pts** Enum constants must be ALL CAPS

QUESTION 4

Polymorphism 18 pts

4.1 Animal a.myName() 3 / 3

✓ + **1 pts** **Wrote "Compiles"**

✓ + **2 pts** **Correct output (I am a Dog named Dog)**

+ **0 pts** Incorrect

4.2 Object d.myName 3 / 3

✓ + **3 pts** **Wrote "Does not compile"**

+ **0 pts** Incorrect

4.3 ((Animal)o2).myName() 3 / 3

✓ + **1 pts** **Wrote "Compiles"**

✓ + **2 pts** **Correct output (I am a Dog named Dog)**

+ **0 pts** Incorrect

4.4 Dog d.myName() 3 / 3

✓ + **3 pts** **Wrote "Does not compile"**

+ **0 pts** Incorrect

4.5 Animal d.bark() 3 / 3

✓ + **3 pts** **Wrote "Does not compile"**

+ **0 pts** Incorrect

4.6 ((Cat o2)).myName 3 / 3

✓ + **1 pts** **Wrote "Compiles"**

✓ + **2 pts** **Correct output (Exception occurs)**

+ **0 pts** Incorrect

QUESTION 5

5 Bonus 3 / 0

+ **3 pts** Correct - 3 good ways

+ **1 pts** at least one good way

✓ + **3 pts** **2 good ways**

+ **0 pts** no good ways

QUESTION 6

6 Arrays 8 / 8

✓ + **8 pts** **Correct**

+ **2 pts** Has int[] arr / correct type

+ **2 pts** arr = new int[arrayLength]; / proper instantiation

+ **1 pts** i < arrayLength or i < arr.length

+ **1 pts** arr[i]

+ **1 pts** arr[i] = 2 * i;

+ **1 pts** return arr;

+ **0 pts** Incorrect

QUESTION 7

7 Inheritance 18 / 20

+ **20 pts** Correct

✓ + **2 pts** **Fish is concrete (no abstract keyword in class header)**

✓ + **2 pts** **Fish extends SeaAnimal**

✓ + **2 pts** **Fish implements Swimmable**

✓ + **2 pts** **Two double instance variables, weight and length**

✓ + **2 pts** **3-argument constructor present**

✓ + **2 pts** **3-arg constructor properly assigns each instance variable (must use super call for name)**

✓ + **2 pts** **Copy constructor present : Fish(Fish other)**

+ **2 pts** Copy constructor correctly copies variables

✓ + **2 pts** **bubbles() is overriden (must have curly brackets, can leave the body empty)**

✓ + **2 pts** **swim() is overriden (must have curly brackets, can leave body empty)**

+ **0 pts** Incorrect

- **2 pts** Major syntax error

- **1 pts** Variable Shadowing

1 If you don't constructor chain or explicitly do a super call to a valid constructor, there is an implicit super call to a non-existent no-args constructor in the super class

QUESTION 8

8 ArrayList 5 / 10

+ **10 pts** Correct
- ✓ **+ 2 pts** **Correct method header (not including the parameter)**
- ✓ **+ 1 pts** **Method parameter is of type ArrayList<String>**
+ **4 pts** Iterates through the ArrayList without skipping Strings or going out of bounds after removals
- ✓ **+ 1 pts** **Iterates through the ArrayList but potentially skips Strings or goes out of bounds after removals**
+ **2 pts** Checks for and removes any even length Strings (must use get() and length() methods)
- ✓ **+ 1 pts** **Modifies original ArrayList**
- **- 2 pts** Major syntax error
+ **0 pts** Incorrect

QUESTION 9

9 Signature 0 / 0

- ✓ **+ 0 pts** **Signed**
- **- 100 pts** Not Signed

CS 1331 Exam 2 - Watson-Phillips October 23, 2019

Name (print clearly): Ruimi Ma

9-Digit GTID: 903405391 Section (e.g, A1): B5

Problem	Type	Points Possible
1) OOP & Arrays	Multiple Choice/True-False	28
2) Equals	Short Coding	8
3) Enums	Short Coding	8
4) Polymorphism	Short Answer	18
5) Bonus	Short Answer	3
6) Arrays	Fill-In-The-Blank	8
7) Inheritance	Coding	20
8) ArrayList	Short Coding	10
TOTAL		103

Please remember: Any academic misconduct (including, but not limited to, the list below) could result in a 0 (zero) on the exam and/or an F grade in the course:

- Communication with anyone other than a proctor for ANY reason in ANY language.
- Sharing of ANYTHING (e.g. pencils, paper, erasers).
- Writing on paper that is not given to you by a proctor.
- Failure to follow directions given by the proctor.
- Use of cell phones, beepers, handheld computers, calculators, during the exam.
- Using books or other reference material.
- Disruption of the exam setting.
- When you turn in your exam, you will have to show your ID to the TAs before we will accept your exam. It is your responsibility to have your ID prior to beginning the exam.
- **IMPORTANT NOTE:** When time is called, stop writing on the exam! **CONTINUING TO WRITE AFTER TIME IS CALLED WILL RESULT IN A 10 POINT PENALTY!!**
- You are not allowed to leave the exam room and return. If you leave the room for any reason, then you must turn in your exam as complete.
- Extra paper is not allowed. If you have exhausted all space on this test, talk with your instructor. There are extra blank pages in the exam for extra space.
- Style standards such as (but not limited to) use of good variable names and proper indentation is always required. If it is unclear which letters are capital and lowercase, underline the capital letters, ex: <u>S</u>ome<u>N</u>ame. Comments are not required unless a question explicitly asks for them.
- **All code must be written in Java with proper syntax (capitalization matters)! No other programming languages will be accepted.**

By taking this exam, you signify that it is your work and that you have neither given nor received inappropriate help during the taking of this exam in compliance with the Academic Honor Code of Georgia Tech.

Signature: Ruimi Ma.

(You must sign this for your exam to be graded!)

This page is intentionally left blank. Do not write ANY answers here, go to the next page.

(28 pts) 1. Multiple Choice - OOP and Arrays

Answer the following questions by **filling in the circle completely** next to the correct answer. Unless otherwise specified in the question, each multiple choice question will only have **one best, correct answer**.

(3 pts) (a) Which of the following should be true, according to the method contract of hashCode(), if a.equals(b) is true?

- ① `a.hashCode() > b.hashCode()`
- ② `a == b`
- ③ `a.toString().equals(b.toString())`
- [✓ ④ `a.hashCode() == b.hashCode()`]
- ⑤ `a.hashCode() < b.hashCode()`

(3 pts) (b) Which of the following is a property of an interface?

- ① Interfaces can implement other interfaces
- [✓ ② Interfaces can have concrete methods] ✓
- ③ All class variables in interfaces must be public ✓
- ④ Interfaces can't have abstract methods
- ⑤ Interfaces can be instantiated

(3 pts) (c) What are the limitations of for-each loops?

- ① Only works for arrays containing reference types
- ② Changes the array's memory location for (int num: arr)
- [✓ ③ Can't modify the original array]
- ④ Only works for int arrays
- ⑤ Always iterates backwards

(3 pts) (d) Given the following statements that have been executed in the command prompt, which of the following code snippets would correctly obtain the String "cats" if placed in the main method?

```
javac Cat.java
java Cat wow those cats are cute
```

Assume this is the main method header:

```
public static void main(String[] args)
```

- ① args[0]
- ② main[2]
- [✓ ③ args[2]]
- ④ main[0]
- ⑤ None of the above

(3 pts) (e) What happens with the following code:

```
int[] a = {1,2,3};
a[3] = 4;
```

① The array length becomes 4
② 4 is added to the back of the array
③ The 3rd element is changed to 4
④ The 3rd element becomes 7
[✓ ⑤] An exception will occur

(3 pts) (f) Which of these properly describes the **dynamic type** of a variable?

① The type of a variable at compile-time
② The type of a variable after all casts have been made
③ The type of a primitive variable
④ The type of a reference type variable
[✓ ⑤] The type of a variable at run-time

(2 pts) (g) A class can directly extend multiple different classes.

① True
[✓ ②] False

(2 pts) (h) A subclass of an abstract class **MUST** implement all of the abstract class's abstract methods. (Make no assumptions about the subclass)

① True
[✓ ②] False

(2 pts) (i) A variable's static type can be an interface.

[✓ ①] True
② False

(2 pts) (j) The following code will compile without error and run without any exception occurring.

```
Integer i = null;
int j = i;
```

① True
[✓ ②] False

(2 pts) (k) The super keyword allows direct access to **all** methods/fields from the parent class.

① True
[✓ ②] False

(8 pts) 2. Short Coding - Equals

Consider a class called Car that has the following instance variables: a String called *model*, an int called *horsepower*, a double called *wheelSize*, and a boolean called *turbo*. Two Cars are considered equal if:

- They have the same model.
- Their horsepower is the same.
- The value for turbo is the same.

With that in mind, write an equals method for the Car class that overrides the equals method inherited from Object. Use the standard convention as taught in lecture.

Write your answer in the box below:

```
@Override
public boolean equals (Object other) {
        if (this == other) {
          return true;
        }
        if (other instanceof Car) {
          Car temp = ((Car) other;
          if (this.model == temp.model && this.horsepower ==
              temp.horsepower && this.turbo == temp.turbo) {
              return true;
          } else {
              return false;
          }
        return false;
        }
```

(8 pts) 3. Short Coding - Enums

Write an enum called *Day* that has a value for each day of the week (Sunday, Monday, Tuesday, etc). The order of the values does not matter. Be sure to follow code conventions for enum constants. No instance variables for each value are necessary.

Write your answer in the box below:

```
public enum Day {
        SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY,
        FRIDAY, SATURDAY;
}
```

(18 pts) 4. **Short Answer - Polymorphism**

Using the page of code on the final sheet of the exam, determine if the block of code in each question will compile. If the code **DOESN'T** compile, write "Does not compile". If the code **DOES** compile, write "Compiles" along with what will be returned by the method call. If an exception will occur at runtime, also write "Compiles. Exception occurs". You can assume all the blocks of code are independent and do not affect one another.

(3 pts) (a)
```
Animal a = new Dog();
a.myName();
```

Compiles I am a Dog named Dog

(3 pts) (b)
```
Object d = new Dog();
d.myName();
```

Does NOT Compile

(3 pts) (c)
```
Object o2 = new Dog();
((Animal) o2).myName();
```

Compiles I am a Dog named Dog

(3 pts) (d)
```
Dog d = new Animal();
d.myName();
```

Does NOT Compile

(3 pts) (e)
```
Animal d = new Dog();
d.bark();
```

Does NOT Compile

(3 pts) (f)
```
Object o2 = new Dog();
((Cat) o2).myName();
```

Compiles. Exception occurs.

(3 pts) 5. **Bonus - Debugging**

Using the space provided, list 3 different ways to debug code **WITHTOUT** the use of a debugging tool like the one in IntelliJ. Only the first 3 answers will be counted towards this question, any more will be ignored.

- Write a driver class to test if the code works logically correct
- Print out the status of variables after every modification is made.
-

(8 pts) 6. **Fill in the Blank - Arrays**

Fill in the blanks to complete the following method, `doubleArray()`. The method takes in an `int` and returns an `int[]`. The array's values at each index should be double of that index. You can assume that only positive numbers will be passed in as arguments. Below are some example inputs and outputs:

```
doubleArray(4) -> {0, 2, 4, 6}
doubleArray(1) -> {0}
doubleArray(6) -> {0, 2, 4, 6, 8, 10}
```

```
public static int[] doubleArray(int arrayLength) {

    // First, declare and instantiate an array of the desired length.

    _int []_______ arr = _new__int[arrayLength]_______________;

    // Loop through the indices in the array

    for(int i = 0; i < _arrayLength_____________; i++) {

        // Assign the value at the specific index in the array

        arr[___i____] = __2 * i_________________________;

    }

    return __arr_________________;

}
```

(20 pts) 7. Coding - Inheritance

Examine the following class and interface carefully.

```
public abstract class SeaAnimal {

    private String name;

    public SeaAnimal(String name) {
        this.name = name
    }

    public abstract void bubbles();

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name
    }

}

public interface Swimmable {

    public void swim();

    public default boolean canSwim() {
        return true;
    }
}
```

On the following page, create a class called *Fish* that meets the following criteria:

- Must be a **CONCRETE** subclass of *SeaAnimal* and implement *Swimmable*
- Must have two double instance variables called *weight* and *length*
- Must have a 3-argument constructor that takes in values for each instance variable and assigns them.
- Must have a copy constructor
- **NOTE:** If you have to override a method, you may leave the method body blank to save space.

Use this space to write the *Fish* class.

```
public class Fish extends SeaAnimal implements Swimmable {
        private double weight;
        private double length;
        public Fish(String name, double weight, double length) {
              super(name);
              this.weight = weight;
              this.length = length;
        }
       public Fish (Fish other) {
             String tempName;
             tempName = other.getName();
             this.setName(tempName);
             this.weight = other.weight;
             this.length = other.length;
     }
     public void bubbles() {

     }
     public void swim() {
     }
}
```

(10 pts) 8. **Short Coding - ArrayList**

Write a static method called removeEvenStrings() that takes in an ArrayList of Strings and removes all of the Strings with an even number of characters. Here is an example input and resulting ArrayList afterwards:

```
Input: {"apple", "pear", "banana", "dragonfruit"}
Output: {"apple", "dragonfruit"}
```

Write your answer below:

```
public class Foo {
```

```
public static void removeEvenStrings(ArrayList <String> arr) {
        int count=0;
        for(int i=0; i<arr.size(); i++) {
            count = (arr.get(i)).size();
            if (count % 2 == 0) {
                arr.remove(i);
            }
        }
    }
```

```
}
```