CS-1331-O1 Exam 2

Vidit Dharmendra Pokharna

TOTAL POINTS

92 / 101

QUESTION 1

Q1-4: T/F 10 pts

1.1 Q1 **2.5 / 2.5**

✓ **- 0 pts** *Correct, F*

- 2.5 pts Incorrect/missing

1.2 Q2 **2.5 / 2.5**

✓ **- 0 pts** *Correct, F*

- 2.5 pts Incorrect/missing

1.3 Q3 **0 / 2.5**

- 0 pts Correct, F

✓ **- 2.5 pts** *Incorrect/missing*

1.4 Q4 **2.5 / 2.5**

✓ **- 0 pts** *Correct, T*

- 2.5 pts Incorrect/missing

QUESTION 2

2 Q5: Object class methods **3 / 3**

✓ **- 0 pts** *Correct: toString()*

- 1.5 pts Circled other choices in addition to toString()

- 3 pts Missing/incorrect

QUESTION 3

3 Q6: Encapsulation of static variable **4 / 4**

✓ **- 0 pts** *Correct, private static double*

- 4 pts Incorrect/missing

QUESTION 4

4 Q7: Visibility modifiers **0 / 3**

- 0 pts Correct, public protected default private

- 1 pts 2 modifiers incorrect

- 2 pts 3 modifiers incorrect

✓ **- 3 pts** *Incorrect/missing*

QUESTION 5

5 Q8: Inheritance hierarchy **3 / 3**

✓ **- 0 pts** *Correct, Object class in java.lang package*

- 1.5 pts Incorrect class name

- 1.5 pts Incorrect package name

- 1 pts Any extra words/characters/notations on an otherwise correct answer

- 0.5 pts Capitalization error

- 3 pts Incorrect/missing

QUESTION 6

6 Q9: Class header **4 / 4**

✓ **- 0 pts** *Correct*

`public final class B extends A`

- 1.5 pts Missing `extends A`

- 1.5 pts Missing `final`

- 1 pts Any other missing/extra keywords or minor errors such as incorrect order of modifiers. - Note: Default visibility is OK for this question

- 4 pts Incorrect

QUESTION 7

7 Q10: Math.random() 3 / 3

✓ **- 0 pts** *Correct, sample solution*

`myVar = (int) (Math.random() * 4) + 2;` (set 2 uses + 3)

- 1 pts Range of values off by more than one digit

- 1 pts Missing cast to int

- 0.5 pts Incorrectly calls Math.random() in any way, such as passing in parameters

- 0.5 pts Missing assignment to local variable

- 0.5 pts Partial: Range of values off by one digit, otherwise correct

- 3 pts Incorrect/missing

QUESTION 8

8 Q11: Superclass header 4 / 5

✓ **- 0 pts** *Correct, sample solutions (two versions):*

`public abstract class Device {
private String id;
private int minutesUsed;
private double powerConsumption;
private static int quantity; // = 0 optional
}`

`public abstract class Person {
private String name;
private int age;
private double happiness;
private static int population; // = 0 optional
}`

Class header

- 0.75 pts Missing public

- 1 pts Missing abstract

- 0.75 pts Missing class

- 0.5 pts Error in order of modifiers

Variable declarations

✓ **- 1 pts** *Not all variables are declared private or declared the correct type*

- 1 pts Static variable missing static modifier

- 5 pts Incorrect/missing

QUESTION 9

9 Q12: Superclass constructor 5 / 5

✓ **- 0 pts** *Correct*

- 1 pts Error in constructor header

- 1 pts Error in constructor parameters, such as types and order

- 2 pts Missing `this` keyword to assign instance variables

- 1 pts Missing incrementation of static variable

- 0.5 pts Minor syntax error

- 5 pts Missing/incorrect

QUESTION 10

10 Q13: Superclass toString() 5 / 5

✓ **- 0 pts** *Correct*

toString() method header

- 1 pts Missing return type String

- 1 pts No-param

- 1 pts Has static keyword

- 1 pts Does not return correct String for either case

- 1 pts Incorrect formatting of output(s)

- 1 pts Method does not return the appropriate value (includes printing rather than return)

- 0.5 pts Minor syntax error

- 5 pts Missing/incorrect

QUESTION 11

11 Q14: Superclass equals() 5 / 5

✓ **- 0 pts** *Correct*

- 1 pts Error in header

- 1.5 pts Missing null check (explicit or implicit via instanceof)

- 1 pts Missing/incorrect downcast

- 1.5 pts Does not return correct result whether devices are equal

- 0.5 pts Does not use equals() to compare String typed instance variables

- 5 pts Incorrect/missing

QUESTION 12

12 Q15: Superclass abstract method 4 / 5

- 0 pts Correct, sample solution:

`public abstract void connectTo(Device other);`

or

`public abstract void greet(Person other);`

✓ **- 1 pts** *Missing semicolon*

- 1.5 pts Missing abstract

- 1 pts Incorrect return value

- 1.5 pts Incorrect parameter type

- 5 pts Incorrect/missing

- 0.5 pts Includes curly braces with semicolon

- 0.5 pts Missing public modifier

QUESTION 13

13 Q16: Superclass method 5 / 5

✓ **- 0 pts** *Correct*

Method header

- 1 pts Missing void keyword

- 1 pts No or incorrect param

- 1 pts Has static keyword

- 2 pts Method does not correctly increase required variable

- 2 pts Method does not correctly determine whether the method should print. Note: ignore the case where the value is exactly the minimum (treat >= and > or <= and < as the same)

- 0.5 pts Minor syntax error

- 5 pts Missing/incorrect

QUESTION 14

14 Q17: Superclass getters and setters 5 / 5

✓ **- 0 pts** *Correct*

- 2.5 pts Incorrect getter (excluding visibility)

- 0.5 pts Incorrect visibility modifier for getter

- 2.5 pts Incorrect setter (excluding visibility)

- 0.5 pts Incorrect visibility modifier for setter

- 1 pts Incorrect capitalization for either method

- 0.5 pts Minor syntax error

- 5 pts Incorrect / Missing

QUESTION 15

15 Q18: Superclass static getter 5 / 5

✓ **- 0 pts** *Correct, sample solution (use variable from set):*

`public static int getPopulation() / getQuantity() { return population / quantity; }`

- 2.5 pts Incorrect method header (including required static keyword, excluding visibility)

- 0.5 pts Incorrect visibility modifier for getter

- 2.5 pts Returns incorrect value

- 0.5 pts Minor syntax error (other than an extra } to close class, this is the last part of the class)

- **5 pts** Incorrect/missing

QUESTION 16

16 Q19: Subclass header 5 / 5

✓ **- 0 pts** *Correct*

Example Answer:

`public class Student extends Person {

private int credits;

}`

Alternate Answer:

`public class Computer extends Device {

private int screenSize;

}`

- **1.5 pts** Header doesn't have "class Student/Computer"
- **1.5 pts** Header doesn't have "extends Person / Device"
- **0.5 pts** Incorrect visibility for class
- **0.5 pts** Extra keyword in class header
- **2 pts** Doesn't have required instance variable
- **0.5 pts** Incorrect visibility for instance variable
- **5 pts** Incorrect/Missing

QUESTION 17

17 Q20: Subclass constructor 5 / 5

✓ **- 0 pts** *Correct*

- **1 pts** Incorrect constructor visibility or name
- **1 pts** Constructor parameters
- **0.5 pts** Constructor parameters do not match instance variables names
- **2 pts** Incorrect constructor chaining with super
- **0.5 pts** Doesn't use this for assigning the instance variable of the class
- **0.5 pts** Minor syntax error
- **5 pts** Incorrect / Missing

QUESTION 18

18 Q21: Subclass chained constructor 5 / 5

✓ **- 0 pts** *Correct*

Header

- **0.5 pts** Missing/Incorrect parameters
- **0.5 pts** Missing/Incorrect parameter types
- **1 pts** Incorrect use of constructor chain
- **1.5 pts** Tries to access variables in parent without super()
- **1.5 pts** Does not set the variable that does not take a parameter
- **0.5 pts** Minor syntax error
- **5 pts** Missing/Incorrect

QUESTION 19

19 Q22: Subclass toString() 5 / 5

✓ **- 0 pts** *Correct*

- **1.5 pts** Incorrect header (excluding visibility)
- **0.5 pts** Incorrect visibility modifier for method
- **1 pts** Doesn't call super.toString correctly
- **0.5 pts** Doesn't use the value from super.toString
- **1 pts** Incorrect String calculated (note that the partial credit alternative may use getters and conditionals to calculate it)
- **1 pts** Doesn't return the calculated String
- **0.5 pts** Minor syntax error
- **5 pts** Incorrect / Missing

QUESTION 20

20 Q23: Abstract method

implementation 3.5 / 5

- **0 pts** Correct
- **1.5 pts** Incorrect method header (excluding visibility)
- **0.5 pts** Incorrect visibility for header

✓ - **1.5 pts** *Doesn't use getter for name/id*

- **1 pts** Incorrect String calculated
- **1 pts** String not printed
- **0.5 pts** Minor syntax error
- **5 pts** Incorrect/Missing

QUESTION 21

21 Q24: Object instantiation 5 / 5

✓ - **0 pts** *Correct*

- **1 pts** Incorrect header for main method (excluding visibility)
- **0.5 pts** Incorrect visibility for main
- **1 pts** Incorrect instantiation of first object
- **1 pts** Incorrect instantiation of second object
- **1.5 pts** No call to connectTo or greet from first object to second
- **1 pts** Doesn't print objects to console (not that calling toString is unnecessary)
- **0.5 pts** Minor syntax error
- **5 pts** Incorrect/Missing
- **0.5 pts** Doesn't instantiate objects with different constructors or specified values

QUESTION 22

22 Q25: Signature 1 / 1

✓ - **0 pts** *Correct*

- **1 pts** Missing

CS 1331 – Exam 2

Fall 2022

Set 2

Name: Vidit Pokharna Section: 01 GTID: 903772087

GT username (i.e. gtg, gth, msmith3, not an alias): vpokharna3

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: viditdpokharna
(you must sign this for your exam to be graded.)

Your grade will be the percentage of points earned out the total points.

Check to make sure your exam has 8 pages. Each page has its number and the total number of pages on the bottom left corner. Your exam will be graded as submitted.

True/False [10pts, 2.5pts each] (approx. 3 minutes)

In each of the blanks below, write "T" if the statement beside the blank is true, "F" otherwise.

1) F You can override a method several times in a single class

2) F Abstract classes can have methods that are final and abstract

3) T Java classes have one primary superclass (assigned explicitly or implicitly) and any number of secondary superclasses

4) T We can access static members of a class in instance methods

Multiple Choice [10pts] (approx. 3 minutes)

5) [3pts] Circle all the methods from the following that exist in all Java classes.

- `new()`
- `equals(Class o)`
- `toString(String formatString)`
- `toString()`
- `main(String[] args)`

6) [4pts] You're writing a House class that has a class variable of type `double` called `averageEnergyConsumption`, which keeps track over time of **the average energy consumption of all houses**. Indicate which of the following **correctly declares and encapsulates the variable** (circle only one):

- `public double averageEnergyConsumption;`
- `private static double averageEnergyConsumption;`
- `private final double averageEnergyConsumption;`
- `private static final double averageEnergyConsumption;`

7) [3pts] Order the visibility modifiers/levels from least (1) to most (4) restrictive.

- `public` 4
- `protected` 3
- `private` 1
- no modifier (default visibility) 2

Short Response [10pts] (approx. 4 minutes)

8) [3pts] Name the class that is at the top of every inheritance hierarchy and indicate to which package it belongs. Remember: capitalization matters.

Class Name: Object

Package: java.lang

Hint for the package: It's the package you don't have to import.

9) [4pts] Write a **class header** for a class B such that B's superclass is A and B cannot be the superclass of any other class.

```
public final class B extends A { }
```

10) [3pts] Write one line of code that assigns a random int value between 3 and 6 (both inclusive) to a local variable myVar, using the Math class.

```
int myVar = (int)((Math.random() * 4) + 3);
```

Class Coding [70pts, 5pts each] (approx. 40 minutes)

Please read carefully:

The remaining questions from this exam are all part of the same problem, divided into many smaller questions. You will be asked to write 2 classes: **`Device`** (an abstract class) and **`Computer`** (a concrete class, and child of Device). Each question will indicate the part and class you will be working on.

- Do NOT write getters and setters unless otherwise indicated. You won't need any that are not indicated (and you cannot use any that are not indicated).
- Do NOT write Javadocs.
- All your constructors and methods should be visible by all classes.
- All your instance variables and class variables should have the strictest visibility modifier.
- When asked for constructors or methods, do not repeat the class header or any other code.
- You should use constructor chaining in all constructors where it can be properly applied.
- Syntax and capitalization matters.

The questions begin in the next page.

11) Write the **class header** and the **variable declarations** (a class without constructors or methods, with correct syntax) for ~~Person~~ Device (an abstract class). Person has the following variables:

- `id` (String)
- `minutesUsed` (int)
- `powerConsumption` (a double)
- `quantity` (this variable, an int, is shared between all instances and keeps track of how many devices exist. Make sure it starts at 0)

```
public abstract class Device {
    protected String id;
    protected int minutesUsed;
    protected double powerConsumption;
    protected static int quantity;
}
```

12) Write a **constructor** for Device. The constructor will take the **`id`, `minutesUsed`**, and **`powerConsumption`** and set all instance variables appropriately. The constructor should also update the quantity variable to indicate that a new device was instantiated. **Your constructor's parameter names should be the same as the instance variables**. Note: even though the class is abstract, this doesn't affect anything on the constructor. You can treat the class as any concrete class in this particular question.

```
public Device(String id, int minutesUsed, double powerConsumption) {
    this.id = id;
    this.minutesUsed = minutesUsed;
    this.powerConsumption = powerConsumption;
    quantity++;
}
```

13) Write a **`toString`** method for Device. It should return "Device [id] used [minutesUsed] minutes with a power consumption of [powerConsumption]." if the minutesUsed is not 0. Otherwise, return "Device [id] with a power consumption of [powerConsumption] hasn't been used."

```
public String toString(){
    if(minutesUsed != 0){
        String str = "Device " + id + " used " + minutesUsed + " ";
        str += "minutes with a power consumption of " + powerConsumption;
        str += ".";
        return str;
    } else {
        String str1 = "Device " + id + " with a power consumption of ";
        str1 += powerConsumption + " hasn't been used.";
        return str1;
    }
}
```

14) Write an **equals** method for Device. Two devices are equal if they have the same id, minutesUsed, and powerConsumption.

```
public boolean equals(Object o) {
  if(!(o instanceOf Device)) {
     return false
  }
  Device other = (Device) o;
  return (id.equals(other.id) && minutesUsed == other.minutesUsed &&
        powerConsumption == other.powerConsumption);

}
```

15) Write an **abstract method** for ~~Person~~ Device called **connectTo**. It receives another Device "other" as parameter, and it doesn't return any value.

```
public abstract void connectTo(Device other) { }
```

16) Write a method for Device called **use**. It receives an int (hours) and increases the minutesUsed of that device (hours times 60). If before the change the device wasn't used for more than 24 hours (minutesUsed less than 1440) but after the change it is, print "Used for at least one day!" to console in its own line.

```
public void use(int time) {
   int original = minutesUsed;
   minutesUsed += (time * 60);
   if(original < 1440 && minutesUsed >= 1440) {
        System.out.println("used for atleast one day");
   }
}
```

17) Write a **getter** and a **setter** in Device for the id.

```
public String getId() {
   return id;
}
```

```
public void setId(String id) {
   this.id = id;
}
```

18) Write a getter in Device for the **`quantity`**.

```
public static int getQuantity() {
    return quantity;
}
```

19) Write the **class header** and the **variable declarations** (a class without constructors or methods, with correct syntax) for Computer (a concrete class, child of Device). The class has the following variables:

- `screenSize` (int)

```
public class Computer extends Device {
    private int screenSize;
}
```

20) Write a **constructor** for Computer. The constructor will take the **id, minutesUsed, powerConsumption**, and **screenSize**, and set all instance variables appropriately. Remember that you cannot directly assign to the variables in Person, and that Student is a child class. **Your constructor's parameter names should be the same as the instance variables.**

```
public Computer(String id, int minutesUsed, double powerConsumption, int screenSize) {
    super(id, minutesUsed, powerConsumption);
    this.screenSize = screenSize;
}
```

21) Write **another constructor** for Computer. The constructor will take only an **`id`**, **`minutesUsed`**, and **powerConsumption**, and set all instance variables appropriately. **Your constructor's parameter names should be the same as the instance variables**. This constructor will be used for **computer with a screen size of 10**.

```
public Computer(String id, int minutesUsed, double powerConsumption) {
    this(id, minutesUsed, powerConsumption, 10);
}
```

22) Write a **`toString`** method for Computer.

- If the device hasn't been used, return "Device [id] with a power consumption of [powerConsumption] hasn't been used. This device is a computer with screen size of [screenSize]."
- Otherwise, return "Device [id] used [minutesUsed] minutes with a power consumption of [powerConsumption]. This device is a computer with screen size of [screenSize]."
- Notice the prefix in both matches the toString from Device. **You must call Device's toString and use its returned value for full credit.**
    - For this question, you are allowed to use getters for id, minutesUsed, and powerConsumption for the partial credit alternative. You don't need to implement the getters for minutesUsed and powerConsumption.

```
public String toString() {
    if (minutesUsed == 0){
        return (super.toString() + " this device is a computer screen with screen
          size of " + screenSize + ".");
    } else {
        return (super.toString() + " this device is a computer screen with
         screen size of " + screenSize + ".");
    }
}
```

23) Write a concrete implementation of the **connectTo** method for Computer (it receives a Device "other" as parameter and doesn't return a value):

- Print "Connecting computer [id] to device [other's id]" on its own line.

```
public void connectTo(Device other) {
    System.out.println("Connecting computer " + id + " to device " + other.id);
}
```

24) Write a **main** method for Computer. It should create one computer with a screen size of 10 and one with a screen size of 20 (any values are other variables are fine, as long as they are valid), using the two different constructors from Computer. Then, the first computer should connect to the other. Finally, print the String representation of both objects to the console (each on its own line).

```
public static void main(String[] args) {
    Computer one = new Computer("12", 12, 12);
    Computer two = new Computer("13", 13, 13, 20);
    one.connectTo(two);
    System.out.println(one);
    System.out.println(two);
}
```

25) [1pt EXTRA CREDIT] Make sure your exam has all the pages. Sign below to confirm your exam has all 8 pages.

Vidit Pokharne

Check to make sure your exam has 8 pages. Each page has its number and the total number of pages on the bottom left corner. Your exam will be graded as submitted.