CS 1331 Exam 2b - Watson-Phillips March 1, 2019

Name (print clearly): _____

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

Problem	Type	Points Possible
1) Foundation	Multiple Choice, T/F, Fill in	20
2) Design and Inheritance	Coding	26
3) Code Analysis - Instance and Static Data	Tracing	15
4) Code Debugging	Tracing	12
5) Code Analysis - Polymorphism	Tracing	11
6) `equals()`, `toString()` and copy constructor	Coding	16
Bonus		2
TOTAL		100 +2

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.
- 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.
- Notes, books, calculators, phones, laptops, smart watches, headphones, etc. are not allowed.
- 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: SomeName
- Comments are not required unless a question explicitly asks for them.

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: _____

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

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

(20 pts) 1. **Multiple Choice, T/F, Fill in the Blank**

Answer the following questions by filling in the circle **completely** next to the correct answer. For fill in the blank questions, write your answer on the line provided.

(2 pts) (a) Given the code below, what does Java insert in the space indicated by the comment?

```
1     public class SomeClass {
2          public SomeClass(int myInt) {
3             // what gets inserted here?
4              x = myInt;
5          }
6     }
```

① `super(myInt);`

② `super(); this();`

③ `this(); super();`

④ `super();`

(2 pts) (b) **T/F:** A sub-class of an abstract class must override all methods defined in the abstract class.

① True

② False

(2 pts) (c) The `Animal` class is `abstract`. `Bird`, `Dog` and `Cat` extend `Animal`. Which of the following is **NOT** legal in Java? (mark all that apply)

① `Animal a = new Cat();`

② `Cat c = new Animal();`

③ `Animal a = new Animal();`

④ `Cat c = new Cat();`

(2 pts) (d) **T/F:** The concept of **overriding** is when a sub-class provides a method with the same name but different parameters as its parent (i.e. it has a different signature).

① True

② False

(2 pts) (e) Is line 3 given below **legal**?

```
1      Double d1 = 10.5;
2      Double d2 = 12.1;
3      if (d1 > d2) {
4          System.out.println("d1 greater than d2");
5      }
```

① True

② False

(2 pts) (f) Given two `Double` objects named `value1` and `value2`, how would you add them together and save the result in a third (primitive) variable called `value3`?

① `double value3 = value1.add(value2)`

② `double value3 = Double.add(value1, value2)`

③ `double value3 = value1 + value2`

(2 pts) (g) **T/F:** If you do not write a constructor for a class, Java will supply a default constructor that has parameters for all instance data in the class.

① True

② False

(2 pts) (h) What visibility modifier should you use on the members of a class so that they are not accessible to another class in a different package, but are accessible to ANY **subclasses** in ANY package?

① public

② private

③ protected

④ final

(2 pts) (i) `Circle` is a sub-class of `Shape`. What is the *static* type of the `circle` variable?

```
Shape circle = new Circle();
```

Answer: _____

(2 pts) (j) `Square` is a sub-class of `Shape`. What is the *dynamic* type of the `square` variable?

```
Shape square = new Square();
```

Answer: _____

(26 pts) 2. **Design and Inheritance**

Implement a sub-class of `Plant` called `Flower` that meets the criteria below. Write your answer on the page provided.

(2 pts) (a) Correctly implements a class called `Flower` that is a sub-class of `Plant`.

(3 pts) (b) Instance data to store if the `Flower` is blooming as a `boolean` called `blooming` and another variable of type `String` called `color`.

(3 pts) (c) A no-argument constructor (`Flower()`) that calls another `Flower` constructor with default values (you decide the values).

(5 pts) (d) A `Flower` constructor that takes in values for **ALL** instance data (including those inherited) and utilizes the parent's constructor to initialize instance data where possible and takes care of the remaining data within the child constructor.

(7 pts) (e) Overrides the `grow` method from `Plant` to only grow if the `Flower` is not blooming AND having a height less than `MAX_HEIGHT`. `Flower`s not meeting these conditions simply return 0. Hint, use the parent's `grow()` method if possible.

(6 pts) (f) Implements a `bloom(int days)` method to: set `blooming` to `true`, print out "blooming..." the number of days passed in, and then sets `blooming` to `false`. For example, if `myFlower.bloom(5)` is called then `blooming...` would be printed 5 times, one per line.

```
1  public abstract class Plant {
2      protected static final int MAX_HEIGHT = 50;
3      protected final int rate;
4      protected int height;
5
6
7      public Plant() {
8          rate = 20;
9      }
10
11     public Plant(int rate, int height) {
12         this.rate = rate;
13         this.height = height;
14     }
15
16     protected int grow (int time) {
17         int amountGrown = time * rate;
18         height += amountGrown;
19         return amountGrown;
20     }
21
22     abstract void bloom(int numDays);
23 }
```

```
//part a
public class Flower _____________________________________ {
    //instance data (part b)









    // no argument constructor (part c)
    public Flower(){









    }
    // constructor that takes in all values (part d)
    public Flower(
```

```
    // Your grow method goes here (part e)
    @Override









    // Your bloom method goes here (part f)









}
```

(15 pts) 3. **Code Analysis**

Given the following code, what is printed? This is not a trick question, this code is valid and compiles. Put your answer in the box at the bottom of the page.

```
1  public class Chair {
2      protected String color;
3      protected static int numLegs;
4      private static int numChairs;
5      public Chair() {
6          this("Blue", 5);
7      }
8      public Chair(String color, int numLegs) {
9          this.color = color;
10         this.numLegs = numLegs;
11         numChairs++;
12     }
13     public String displayChair() {
14         return "color == " + this.color + "; numLegs == " + numLegs;
15     }
16     public static void main(String[] args) {
17         Chair c1 = new Chair("Green", 5);
18         System.out.println(c1.displayChair());
19         Chair c2 = new Chair();
20         System.out.println(c2.displayChair());
21
22
23         c1.numLegs = 4;
24         System.out.println(c2.displayChair());
25
26         c1 = new Chair();
27
28         System.out.println(c1.equals(c2));
29         System.out.println(numChairs);
30     }
31 }
```

(12 pts) 4. **Code Debugging**

There are some errors in the code given below. Fill in the table with the missing information. Each row should indicate if there is a compile time exception or logic error (including a runtime exception), on which line(s) the error occurs and an explanation (just listing the exception will not get you full credit, tell me why). Assume that any compile time errors are resolved in order to evaluate the remaining code.

```
1  public class ZooSim {
2       private Animal[] animals;
3       public Double numAnimals;
4
5       public ZooSim(int numAnimals) {
6           animals = new Animal[numAnimals];
7       }
8       public static void main(String[] args) {
9           ZooSim sim = new ZooSim();
10          ZooSim sim2 = new ZooSim(20);
11          ZooSim sim3 = new ZooSim(5);
12          double numAnimals = sim2.numAnimals;
13          for(int i = 0; i < sim2.animals.length; i++) {
14              System.out.println(sim3.animals[i]);
15          }
16      }
17 }
```

Line Number	Error type	Explanation
	Compile Exception	
12	Logic Error (Runtime Exception)	
13-14	Logic Error (Runtime Exception)	

(11 pts) 5. **Polymorphism**

Given the code below, what is the output when the main method in `Parent` is run? Answer the question in the provided box on the next page.

```
public class Parent {
    protected String name = "Mom";
    protected String myNameIs() {
        return "I am the Parent! My name is: " + getName();
    }
    public String getName() {
        return name;
    }
    public static void main(String[] args) {
        Parent[] myArray = new Parent[3];
        myArray[0] = new Child("Child 1");
        myArray[1] = new Parent();
        myArray[2] = new FavoriteChild("Mom's favorite");
        for(Parent myObject: myArray)
            System.out.println(myObject.myNameIs());
    }
}
---- New class in a new file----
public class Child extends Parent {
    public Child(String name) {
        this.name = name;
    }
    @Override
    protected String myNameIs() {
        return "I am " + getName();
    }
}
---- New class in a new file----
public class FavoriteChild extends Parent {
     public FavoriteChild(String name) {
          this.name = name;
     }
     protected String myNameIs() {
          return "Of course... I am " + getName();
     }
     public String getName() {
          return "the favorite child";
     }
}
```

Answer for question 5

(16 pts) 6. **equals, toString, and copy constructors**

Given the following class definition, implement the following:

(6 pts) (a) Copy constructor that deep copies all of the class's data, assume all the classes of the instance variables have their own deep copy constructors (i.e. the `Engine` class has a deep copy constructor that takes in an `Engine` object)

(7 pts) (b) `equals` method that overrides `Object`'s `equals` method and only returns true if all fields are equal.

(3 pts) (c) `toString` method that returns a `String` with all of the `LawnMower`'s data

You can assume that any classes included already have their own properly implemented `equals`, `toString` and copy constructor methods.

```
public class LawnMower {
    protected int year;
    protected String brand;
    protected Double hp;
    protected Engine engine;

    public LawnMower(int year, String brand, Double hp, Engine engine) {
        this.year = year;
        this.brand = brand;
        this.hp = hp;
        this.engine = engine;
    }
```

<continued on next page>

```
// Copy constructor
public LawnMower(LawnMower mower) {









}
// equals method
public boolean equals(Object o) {









}
```

<continued on next page>

```
    // toString method
    public String toString() {









    }
}
```

(2 pts) 7. **Extra Credit Bonus**

List the 4 software development activities or categories discussed in class. Recall the some require more creativity than others, some require client interaction, some benefit from diagrams and one always happens last.