Midterm Examination [redacted] — June 2023

CSC207H1Y — Software Design

Robert Wu
rupert.wu@utoronto.ca

Department of Computer Science
University of Toronto

June 21, 2023

Question 1. Version Control [11 MARKS]

(a) [2 MARKS] Briefly give two reasons for using version control like `git`.

(b) [2 MARKS] What is a non-trivial merge conflict between two branches? Be precise.

(c) [2 MARKS] Name two common ways that merge conflicts are resolved.

(d) [5 MARKS] After cloning, `Main.java` is modified to support the French language. Use a typical `git` workflow in command-line to update only `Main.java` in the remote repository. Include a command to check tracked/staged file changes before staging. You're not required to use every line of space.

```
# suppose repository has been cloned
$ ls
src test README.md
# files including Main.java have been modified.
$ git
$ git
$ git                                    support"
$ git
```

Question 2. Basics of Java [14 MARKS]

(a) [3 MARKS] What's different between compiled and interpreted languages? Give an example of each.

(b) [1 MARK] How would you describe Java on this spectrum? What does it run on?

(c) [1 MARK] What's the difference between a function and a method?

(d) [4 MARKS] Declare and initialize a `float f`, `boolean b`, `long l`, and `char c` in ascending order of their memory sizes (i.e. how many bits to represent each type) with any valid values. Use the code block on the right.

```
=
=
=
=
```

(e) [3 MARKS] Program `Bill` doesn't compile. Point out **three** *different* potential compile errors.

```
class Bill {
    private double rate = 0.13;
    private double tax(double principle) {
        return (1 + rate) * principle;
    }
    public static void main(String[] args) {
        float fee = 5.78;
        float fee = tax(fee);
    }
}
```

(f) [2 MARKS] Briefly give one reason for and one reason against using `float` instead `double`.

Question 3. Object-Oriented Programming (OOP) [13 MARKS]

Class `Constant` (right) always returns a. We want to write a program to model linear functions $f(x) = bx + a$ and quadratic functions $g(x) = cx^2 + bx + a$, each with its own `compute()` method. Look closely at what they have in common.

```
class Constant {
    private double a;
    public Constant(double a) { this.a = a; }
    public double compute(double x) { return a; }
}
```

(a) [6 MARKS] Use chained inheritance to write `Linear` and `Quadratic` while reusing code when possible. Follow best practices when overriding methods and shadowing variables.

```
class Linear [redacted] {
    [redacted]
}
```

```
class Quadratic [redacted] {
    [redacted]
}
```

(b) [2 MARKS] So far, functions descend from `Constant` but really, we only need a type that provides the `compute(double x)` behaviour. Write such a type `OneVar` and adapt `Constant`'s header to it.

```
[redacted] OneVar {
    [redacted]
}
```

```
[redacted] Constant [redacted] OneVar {
    [redacted]
}
```

(c) [5 MARKS] Construct a function of each of the above classes given `a`, `b`, `c`. Load them into a single `ArrayList` that stores any arbitrary functions. Finally, compute and print their their values at x.

```
import java.util.ArrayList;
public class PolyInheritance {
    private static double a = 1.73, b = 0.2, c = -2.88, x = 82.3;
    public static void main(String args[]) {
        [redacted]
    }
}
```

[redacted]

Question 4. Exceptions & Errors [10 MARKS]

(a) [3 MARKS] Explain in detail the difference between a `RuntimeException` and a regular `Exception`.

[redacted]

(b) [5 MARKS] Wrap the following file-opening code in an exception handling structure.

```
String filePath = "/home/rhubarb/file.md";
BufferedReader reader = new BufferedReader(new FileReader(filePath)); // FileNotFoundException?
reader.close() // IOException?
```

Consider also,

- Any exception `e` should be handled with `e.printStackTrace()`.
- `reader` is not guaranteed to be non-`null` at all times.
- Treat `reader.close()` as tear-down/clean-up code.
- Exception handling blocks can be nested.

```
String filePath = "/home/rhubarb/file.md";
// start here
[redacted]
```

(c) [2 MARKS] If the above code were in a method, and you don't want to handle these exceptions here, **briefly** describe how you would modify your code. Be sure to use the exact Java keywords.

[redacted]

Question 5. Design Principles [8 MARKS]

(a) [2 MARKS] What is persistence? Give two examples of mechanisms.

(b) [2 MARKS] Name the five design principles of SOLID.

(c) [2 MARKS] Explain the "D" and one of its benefits in two sentences.

(d) [2 MARKS] Name two typical violations of CA.

Question 6. Design Processes I [8 MARKS]

Consider an airline management system that tracks passengers and flights. Passengers have personal information, customer records (including loyalty points), and booking information with respect to flights. For flights, the system records information about aircraft, schedules, prices, luggage destination/storage locations, pilots and staff. When a customer wants to purchase a ticket, the airline should be able to calculate the fare based on flight details and customer points. When delaying or cancelling flights, the airline should be able to look up flight details, make modifications, and notify customers.

(a) [3 MARKS] Identify six likely entity classes. [redacted]

(b) [1 MARK] Identify the two obvious use-case classes. [redacted]

(c) [4 MARKS] Draw a CRC card for each use-case class, including relationships with all applicable entity classes you identified. [redacted]

Question 7. Design Processes II [10 MARKS]

Refer to `Kitchen.java` on the last page that models a restaurant kitchen and answer the following questions. Anywhere you see `/* implementation */` or `/* definition */` is code that is hidden to simplify this question; don't worry about it.

(a) [1 MARK] Why might some subclasses of `Worker` have constructors with different parameters?

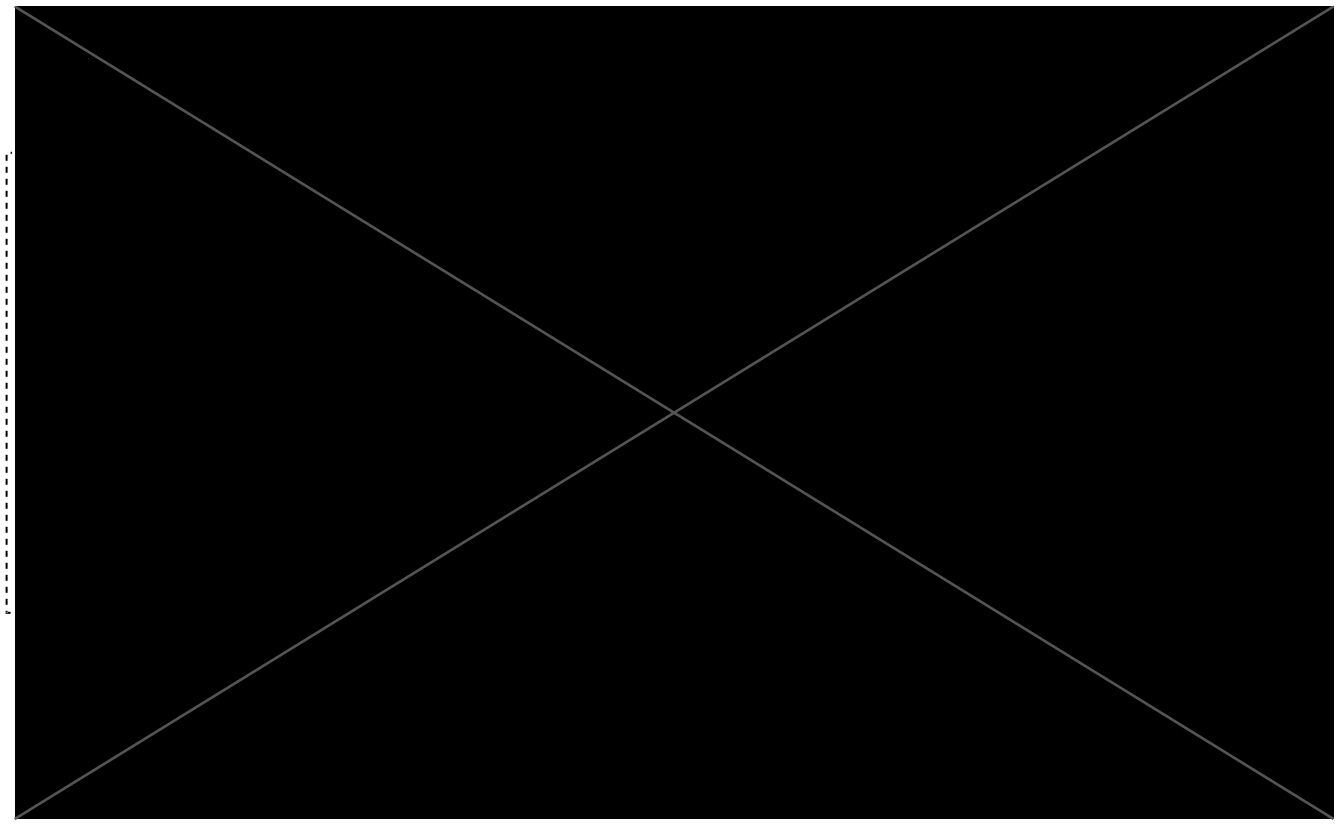
[redacted]

(b) [2 MARKS] According to code, does `Chef` cook? Explain why or why not.

[redacted]

(c) [7 MARKS] Draw a UML diagram to model `Kitchen.java`. Do not include constructors. You may tear off the last page containing `Kitchen.java` if that helps; just make sure to include it in your submission, especially if you need the blank page (backside) for your answers.

[redacted]



Kitchen.java

```
/* exclude boxes for these classes in your UML; just indicate type dependencies */
class Recipe { /* definition */ }
class Ingredient { /* definition */ }
class Dish { /* definition */ }
enum CuisineSpecialty { /* definition */ } // treat this as a primitive type

/* include boxes for these classes in your UML */
abstract class Worker {
    protected String name;
    protected float hourly;
    public Worker(String name, float hourly) { /* implementation */ }
    public String getName() { /* implementation */ }
}
interface Cooking {
    public Dish prepare(Recipe recipe, ArrayList<Ingredient> ingredients);
}
interface Procurement {
    default public ArrayList<Ingredient> order(ArrayList<String> names) { /* implementation */ }
}
class Cook extends Worker implements Cooking {
    public Cook(String name, float hourly) { /* implementation */ }
    public Dish prepare(Recipe recipe, ArrayList<Ingredient> ingredients) { /* implementation */}
}
class Chef extends Cook implements Procurement {
    private CuisineSpecialty specialty;
    public Chef(String name, float hourly) { /* implementation */ }
}
class LineCook extends Cook implements Cooking {
    public LineCook(String name) { /* implementation */ }
}
class Server extends Worker {
    public Server(String name) { /* implementation */ }
}
class Manager extends Worker implements Procurement {
    public Manager(String name, float hourly) { /* implementation */ }
}
class Owner extends Worker {
    public Owner(float hourly) { /* implementation */ }
}
```

You may tear off this page to help you complete the UML question; just make sure to include it in your submission, especially if you need the blank page (other side) for your answers.