# University of Windsor logo Midterm 2021F - COMP 3220

# Object-Oriented Software

# Analysis and Design

# School of Computer Science

# University of Windsor

**November 1st, 2021**

**Total: 100 marks – SUBMIT INDIVIDUALLY VIA BLACKBOARD**

**Long Answer (40 marks total - 20 marks each):**

1. Provide an example of an Information System, and describe it in terms of each of the following properties:
   1. Inputs
   2. Outputs
   3. Purpose
   4. Boundary
   5. Subsystems
   6. Interfaces
   7. Control
   8. Emergent Properties
   9. Environment

* Banking system is a decent example of what we call **Information Systems.** Banking system consist of Customers, Manager, Staff, Accounting, and many more subsystems. Let’s talk about each of the above properties to better understand Banking System.
  1. Example of **Inputs** could be opening a new bank account, deposit money to the account, editing customer information, login id and password for each employee and manager.
  2. **Outputs** contains withdrawing money, accessing the bank account successfully, Paying salary to the staff and manager.
  3. **Purpose** of the banking system is very clear. To provide financial services for the customers, operating money exchange and offering loan, providing deposit/withdrawal system.
  4. **Boundary :** Banking system is only useful in financial services.
  5. **Subsystems** are as follows:
     + Customer
     + Staff
       - Employee
       - Manager
     + Accounts
       - Savings
       - Checking
     + Withdrawal
       - By ATM
       - By Cheque
       - Sending Money
     + Deposit
       - E-Transfer
       - Manual Deposit
  6. **Interfaces** as follows:
     + Account authorization
     + Payment process
     + E-transfers
     + Analytics
     + Online Banking platform
     + Monthly Statements
  7. **Controls :** 
     + Policies
  8. **Emergent Properties**
     + **A**
  9. **Environment:** 
     + Financial Information System

1. Create a communication diagram given the following written description of a use case in a staff management system for a local tech company wherein a manager can modify its employee’s details. Assume the manager is already at a “Change Employee Details” screen on the application. On this screen, the manager first clicks on a button that says “Select Employee”. The manager can then type the employee’s name in a text box (which will autocomplete). Alternatively, the manager can search for the employee by employee type (engineer, software developer, sales, production). Once an employee has been selected, the employee’s details are shown on the screen. The manager can then make any changes they like, and then exit the screen to save the current details. The editable properties of the employee are name, hourly wage, type, phone number, and address.

Graphical user interface

Description automatically generated with medium confidence

**Short Answer (40 marks total):**

1. **(5 marks total)**
   1. **Describe** the difference between a **deep copy** and a **shallow copy**. (2 marks)

|  |  |
| --- | --- |
| **Deep Copy** | **Shallow Copy** |
| Deep copy constructs an object with the same values as the original object | Shallow Copy stores reference of the memory address of the original object |
| Modifying the values in the new/copied object won’t change the original object in any way | Modifying the values in the new/copied object will reflect on the original object as well |
| Deep copy takes time to copy an object | Shallow Copy is faster compare to deep copy |

* 1. **Describe** what tests should be done to ensure a deep copy was properly implemented. (3 marks)
     1. All the values of the original object should equals to the all the values in the new/copied object.
     2. Both object shouldn’t share same memory address, i.e., changing the values in one object shouldn’t change the value in the new/copied object.

1. **(5 marks)** Consider the very simple Cat class, and the corresponding unit test case TestCat, specified below.

**Class Cat:**

**public** **class** Cat {

**public** String name; // Changing to public

**public** **int** age; // Changing to public

**public** Cat(String name, **int** age) {

**this**.name = name;

**this**.age = age;

}

// Override method for equals returns true if both obj has same values

@Override

**public boolean** equals(Object obj){

if((this.name.equals(obj.name)) && (this.age == obj.age))

**return** true;

**return** false;

}

}

**Class TestCat:**

**import** **static** org.junit.jupiter.api.Assertions.\*;

**import** org.junit.jupiter.api.Test;

**class** TestCat {

@Test

**void** testEquality() {

Cat a = **new** Cat("Mr. Fluffy", 1);

Cat b = **new** Cat("Mr. Fluffy", 1);

*assertEquals*(a, b);

}

}

* 1. The assertEquals call in testEquality of TestCat fails. **Describe why.** (3 marks)
     1. The assertEquals failed because both objects that we created cat::a and cat::b are two different objects, with different memory addresses, That is why, when checking if both objects are same using assertEquals, we get a result that the test case failed.
  2. **Write one method** in Cat that would make assertEquals pass, as it should. (2 marks)
     1. **Check the code given above**

1. **(5 marks)** Explain how data, information, and knowledge are related.
   1. **Data** is a block of related values. When we convert this data into something meaningful or pay attention and redeem something that we can use with a context, Data becomes **Information**. All data is not information but all the information is data. **Knowledge** is related to information in a way that we take actions after getting the information or relating the information with the previously learned information. Knowledge has a context, a clear meaning and some action/reflexes relating to the information.
2. **(5 marks)** Compare and contrast between Waterfall and Extreme Programming methodologies.

|  |  |
| --- | --- |
| **Waterfall** | **Extreme Programming** |
| Waterfall is continuous, it is hard to go to the previous stage. | Extreme Programming is iterative, and have a lifecycle that runs everytime untill the final product is done. |
| Testing in the waterfall can be conducted only at the final stage | Testing can be conduct in every lifecycle to ensure each piece of code is working properly before moving on to the next stage. |
| One time development and harder to find/solve bugs in the final stage | XP has constant involvement and easier to meet customer requirements and solve bugs. |

1. **(10 marks)** Compare, using examples (Java or pseudocode), the concepts of branch coverage, decision coverage, and condition coverage.
   1. **Branch Coverage**

|  |
| --- |
| If (num >= 0 && num <= 0)  Print(“zero”);  Else  Print(“positive or negative”); |
| To insure 100% branch coverage, we have to make sure to test every branch there is in the code. For example, in the above code, there are two branches.. if the condition is true, and if the condition is false. Here, we care less about decision coverage and won’t care about every possible outcome of the if condition. |

* 1. **Decision Coverage**

|  |
| --- |
| If(a > b)  If(a > c)  Return a;  Print(“lol”);  Else  If(b > c)  Return b;  return c; |
| In Decision coverage, we have to check every possible outcome of all the decisions we implemented in the code. For exmple, in the above code, we have first condition if (a > b) has two outcomes, either a > b or a < b.. if a > b then we have another decision and that has two outcomes as well. So in order to test this code in 100 % decision coverage, we have to check for all the posibile test given below   * a > b && a > c * a > b && a < c * a < b && b > c * a < b && b < c |

* 1. **Conditional Coverage**

|  |
| --- |
| If(x && y) print (“hello”);  Else print(“bye”); |
| Conditional coverage means to test different possible combination of the condition and care less about if it tests all the statements in the code. For example, in the above code, we can simply test two possibilities, 1. x = true & y = false, 2. x = false & y = true. Although using only this possibilities doesn’t test every statement, i.e., for both scenario, only else statement is executed. |

1. **(5 marks)** What is meant by “black box” testing? How does this differ from “glass box” testing?
   1. Black box testing generally means testing of the behavaour of the code without knowing the implementation. Black box testing only cares about correctness of output/functionality as seen from the outside rather than worrying about if every possible code path is reachable.
   2. Black box ignores the implementation of the code while glass-box testing makes sure to test every posible outcome/combination to ensure 100% statement coverage.
2. **(5 marks)** Describe, with examples, the relationship between Unified Process **phases** and **disciplines**.
   1. Unified process divides the real world project into 4 phases which are, Inception, Elaboration, Construction, and Transition. In every phase, there’s one or more iterations containing Unified process disciplines for Customer requirements, testing, implementing, delploying and more… These displines and phases iterates through many time to finally create a final product/system with all the user requirements.

**Multiple Choice and (True or False) (1 mark each – 20 marks total):**

1. (True or False) The Agile methodology aims to support all programming with rigorous, detailed documentation.
   1. **True**
2. (True or False) Prototypes should always be developed for implementation in the final product.
   1. **False**
3. (True or False) Model value and complexity are related – as a model’s complexity increases, its value increases.
   1. **False**
4. (True or False) All models are diagrams, and all diagrams are models.
   1. **False**
5. (True or False) One key issue with UML is that the diagrams are not standardized, which means that UML diagrams can be easily misinterpreted.
   1. **False**
6. (True or False) There are five views which can be described using UML – any combination of them may be used in a given project.
   1. **True**
7. (True or False) Background Reading and Document Sampling have different aims and consequently involve different types of documents.
8. (True or False) “Links” and “Associations” are the same in UML – they are edges between nodes in UML diagrams.
   1. **False**

1. (True or False) Multiplicity shows how many instances of a given class can exist in an implementation of an information system.
   1. **True**
2. (True or False) As a software development project progresses, requirements become less and less volatile.
   1. **False**
3. (Multiple Choice) Which of these is **not** an example of an **emergent property** of a system:
   1. The maximum number of users that might simultaneously have an active connection to a server program in a given day.
   2. **A predefined limit to the number of parallel subprocesses allowed in a program.**
   3. The average time it takes for a database management system to process a query.
   4. The most common button clicked on a given screen in a GUI-based information system.
4. (Multiple Choice) Which of the following **is achieved** by testing the following conditional statement with inputs (i=True, j=True) and (i=True, j=False):  
   if i:

If j:

print(“yay”)

else:

print(“nay”)

* 1. 100% decision coverage
  2. **100% statement coverage**
  3. 100% condition coverage
  4. 100% modified condition/decision coverage

1. (Multiple Choice) Which of the following **is not achieved** by testing the following conditional statement with inputs (i=True, j=True) and (i=False, j=False):  
   if i:

If j:

print(“yay”)

else:

print(“nay”)

* 1. **100% decision coverage**
  2. 100% statement coverage
  3. 100% condition coverage
  4. 100% branch coverage

1. (Multiple Choice) In the **elaboration** phase of the Unified Process, which of the following disciplines is used **most frequently**:
   1. **Requirements**
   2. Implementation
   3. Testing
   4. Deployment
2. (Multiple Choice) In the **construction** phase of the Unified Process, which of the following disciplines is used **least frequently**:
   1. Design
   2. Implementation
   3. Testing
   4. **Requirements**
3. (Multiple Choice) Which of the following is **not** a typical focus in the **elaboration** phase of the Unified Process:
   1. Understanding key use cases
   2. Developing prototypes of risky features/subsystems
   3. **Planning the first iteration**
   4. Establishing intermediate milestones
4. (Multiple Choice) Which of the following is an example of **corrective maintenance**:
   1. Adding a new feature to navigate back to the main menu screen from another specific screen.
   2. Adding a new column in a database table and adding the corresponding functionality to work with data in the new column.
   3. Adding new functionality because of a change in how the users want to interact with a given screen.
   4. **Limiting the range of values for a specific variable such that the system no longer crashes if negative values are provided.**
5. (Multiple Choice) Which of the following directly results in a **quality issue** in **Information Systems development**:
   1. **Poorly analyzing the user’s requirements**.
   2. User’s requirements changing over time.
   3. Unforeseen roadblocks in development
   4. External events, like the emergence of COVID 19.
6. (Multiple Choice) Which of the following is an example of **positive feedback**:
   1. A business, wisely investing its revenue such that it can now make more money faster.
   2. **Temperature control using a thermostat.**
   3. The population size of a prey animal, given the population size of its predators.
   4. Shivering, when too cold, to make your body warmer.
7. (Multiple Choice) Which of the following are **not** characteristics of **extreme programming**:
   1. Maximize how much code is written by making minimal documentation
   2. **Thorough requirements analysis occurring at the beginning of a project**
   3. Rigorous testing
   4. Easily adapting to changes in user requirements