Friday xth April 2019, x.00 pm - x.30 pm
(1 hour 30 minutes)

DEGREES OF MSci, MEng, BEng, BSc, MA and MA (Social Sciences)

# COMPUTING SCIENCE 2Y: OBJECT ORIENTED SOFTWARE ENGINEERING 2

**INSTRUCTIONS -READ CAREFULLY**

- **THE MULTIPLE CHOICE QUESTIONS ARE MARKED AS FOLLOWS:**
  - **If you answer correctly, you will be awarded the mark shown**
  - **If you give no answer, you will be awarded 0 marks**
  - **If you answer incorrectly, 1 mark will be taken off your total**
- **For all questions, write your answers as clearly and legibly as possible in the separate exam answer booklet.**

**Answer all questions in Part 1 and Part 2**

**This examination paper is worth a total of 80 marks (Part 1 and 2 each worth 40 marks)**

**The use of a calculator is not permitted in this examination.**
**INVIGILATORS: Question papers should be collected at the end of the exams.**

## Part 1: Multiple Choice Questions [40 Marks]

**1.** Which of the following metrics would you use if you are concerned about the number of test cases you need to fully verify your system? [2]

**(a)** NOC (Number of Children)

**(b)** WMC (Weighted methods per class)

**(c)** CBO (Coupling between Objects)

**(d)** RFC (Response for class)

**(e)** All of the above

**Solution:** All of the above (e)

**2.** You compute the LCOM (Lack of Cohesion Methods) for a class and you get a negative value. What does this mean? [2]

**(a)** It is impossible to have a negative value

**(b)** The software is well designed

**(c)** The software is badly designed

**(d)** It is impossible to determine the quality of software based on LCOM

**(e)** None of the above

**Solution:** The software is well designed (b)

**3.** In Java, what is the minimum value of DIT (Depth of Inheritance Tree)? [2]

**(a)** 0

**(b)** -1

**(c)** 1

**(d)** 2

**(e)** None of the above

**Solution:** 1 (c)

**4.** Consider the formula:

$$A = \sum_{i=1}^{n} c_i$$

where $c_i$ is the complexity of each method in a class and $i$ is the number of methods in the class. Which software metric does $A$ represent? [2]

**(a)** LCOM

**(b)** RFC

**(c)** NOC

**(d)** DIT

**(e)** WMC

> **Solution:** WMC (e)

**5.** Which of the following are subjective complexity factors in a program code? [you may select more than one answer] [2]

**(a)** The amount of code coupling

**(b)** The amount of test cases required to cover the code

**(c)** The cyclomatic complexity of the code

**(d)** The variables and methods naming style

**(e)** Following a design pattern

> **Solution:** (d), (e)

**6.** Which of the following suggests that there is a bug in the software? [you may select more than one answer] [2]

**(a)** The software fails to compile

**(b)** The software fails at runtime

**(c)** The software is implemented using iterative development approach

**(d)** The software is implemented using waterfall development approach

**(e)** End users are finding it difficult to use the software

> **Solution:** (a), (b) and (e) (NOTE from external examiner: (a) or (e) should still receive credit)

**7.** Which of the following approaches can be used to identify bugs in software? [you may select more than one answer] [2]

**(a)** Manual inspection of program code

**(b)** Dynamic analysis of the program code

**(c)** Static analysis of the program code

**(d)** Software testing (whitebox/blackbox testing)

**(e)** Threat Modelling

**Solution:** (a), (b), (c), (d) and (e)

**8.** Say your program has 100 real bugs. Which debugging tool is likely to ensure that the software is shipped with a maximum of 20 bugs? [2]

**(a)** A tool that finds 80 bugs but reports 200 warnings

**(b)** A tool that finds all 100 bugs, but reports 1,000 warnings

**(c)** A tool that flags every program statement as a bug

**(d)** A tool that does not flag any bug

**(e)** None of the above

**Solution:** None of the above (e)

**9.** Which of the following is **not** true about a bug detection system? [2]

**(a)** A bug detection system uses an algorithm that is based on a trade-off between soundness, precision and execution time

**(b)** Bug detection systems may sacrifice soundness for precision

**(c)** A precise bug detection system means that it can generate false positive outputs

**(d)** An unsound bug detection system means that it can generate false negative outputs

**(e)** It is impossible to generate a debugger that captures all forms of bugs irrespective of associated practice.

**Solution:** (c)

**10.** Which of the following is **not** a bug pattern category? [2]

**(a)** Multithreaded correctness

**(b)** Internationalization problems

**(c)** Correctness of the program

**(d)** Pass by value

**(e)** Performance

**Solution:** (d)

**11.** Which of the following is **not** an output from a debugger? [you may select more than one answer] [2]

**(a)** Bug Pattern Code

**(b)** Source Line Number

**(c)** Descriptive Message

**(d)** Bytecode

**(e)** Configuration file

**Solution:** (d), (e)

**12.** Which of the following frameworks cannot be used for bytecode analysis? [2]

**(a)** Eclipse JDT

**(b)** AOP (AspectJ)

**(c)** CGLIB (Byte Code Generation Library)

**(d)** JavaAssist (Java Programming Assistant)

**(e)** BCEL (Byte Code Engineering Library)

**Solution:** (a)

**13.** Which of the following statements is inaccurate about domain modelling? [2]

**(a)** A domain model is a conceptual model of the domain that incorporates both behavior and data

**(b)** A domain model is a set of abstractions that describes an operational context

**(c)** A domain model is used to solve problems related to that domain

**(d)** A domain model is an instrument to foster better stakeholder communication and product quality

**(e)** Java is a suitable programming language for domain modelling

**Solution:** (e)

**14.** Consider the class diagram in Figure 1. Which of the following visibility properties will you assign to attributes in the design?

[2]

**(a)** $+$ public

**(b)** # protected

**(c)** $-$ private

**(d)** $\sim$ package (default)

**(e)** / derived

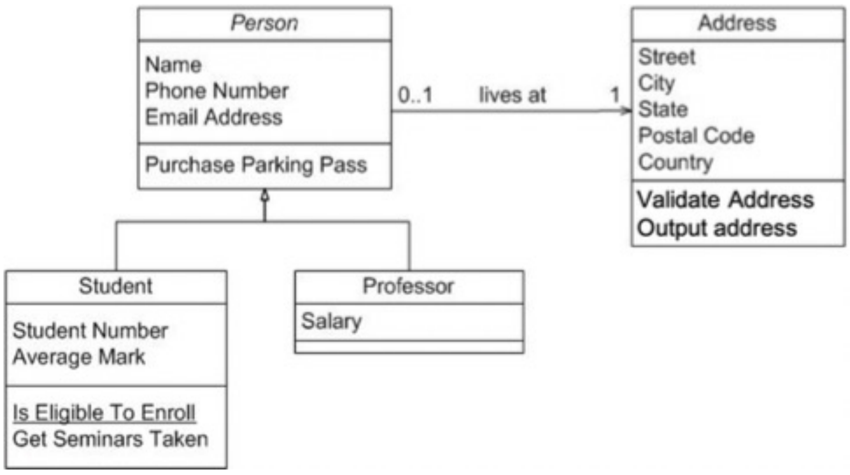

Figure 1: Class Diagram

**Solution:** (c )

**15.** Which statement(s) is incorrect about the design in Figure 1?[you may select more than one answer] [2]

**(a)** A Student is a type of Person

**(b)** A Professor is a type of Person

**(c)** A Person lives at only one Address

**(d)** A Student may not have an Address

**(e)** A Student is the parent of Person

**Solution:** (d), (e)

**16.** Assume you do not want any object instance that is not a Person to purchase a parking pass in Figure 1. What visibility property will you assign to a function that implements this feature? [2]

**(a)** $+$ public

**(b)** # protected

**(c)** $-$ private

**(d)** $\sim$ package (default)

**(e)** / derived

**Solution:** (b)

**17.** What are class diagrams **not** good for? [you may select more than one answer] [2]

**(a)** Discovering algorithmic behavior

**(b)** Checking whether the relationships between objects are too complex, too many in number or simple enough

**(c)** Finding the flow of steps for objects to solve a given problem

**(d)** Understanding the overall control flow of the software

**(e)** Discovering related data and attributes

**Solution:** (a), (c ), (d)

**18.** Which of the following is false about the state design pattern? [2]

**(a)** It encapsulates state into separate classes

**(b)** Each state has a different behaviour

**(c)** It finds the flow of steps for objects to solve a given problem

**(d)** It favours inheritance over composition

**(e)** It keeps a class closed for modifications but open for extension

**Solution:** (d)

**19.** MVC is a combination of which of the following design patterns? [you may select more than one answer] [2]

**(a)** Strategy

**(b)** Observer

**(c)** Visitor

**(d)** Composite

**(e)** Factory

**Solution:** (a), (b), (d)

**20.** Which of the following design patterns can be used to analyse an abstract syntax tree (AST)? [2]

**(a)** Decorator

**(b)** Singleton

**(c)** Visitor

**(d)** Composite

**(e)** Factory

**Solution:** (c)

## Part 2: Essay Questions [40 Marks]

**1.** Consider the `split` function that used to split a string into a limited number of substrings:

```
String[] split(String str, int size)
```

**(a)** Define the partitions, blocks, values and three possible boundary inputs for `split`. [8]

**Solution:** Two marks each for appropriate identification of partitions, blocks, values and appropriate boundary inputs.

1. *Partition*: `size` value
   - $b_1$ = <0
   - $b_2$ = 0
   - $b_3$ = >0

Possible inputs:
- $b_1$ : `size` = -1
- $b_2$ : `size` = 0
- $b_3$ : `size` = 1
- $b_3$ : `size` = MAXINT

2. *Partition*: `str` with length
   - $b_1$ = < `size`
   - $b_2$ = in[`size,size`*2]
   - $b_3$ = > `size*2`

Possible inputs:
- $b_1$ : `str` with length = `size`−1
- $b_2$ : `str` with length = `size`
- $b_3$ : `str` with length = `size*2+1`

etc…

**(b)** How many independently testable features do we have in `split`? [2]

**Solution:** only 1. Since it is only one function

**(c)** During combinatorial testing, meaningless test cases may also generated. List the six steps in Category-Partition Method that helps to mitigate this problem ? [6]

**Solution:**

1. Identify independently testable features
2. Identify categories
3. Partition categories into choices
4. Identify constraints among choices
5. Produce/evaluate test case specification
6. Generate test case from test case specifications

**2.** Suppose that coverage criterion C1 subsumes coverage criterion C2. Further suppose that test set T1 satisfies C1 on program P and test set T2 satisfies C2, also on P.

**(a)** Does T1 necessarily satisfy C2? Explain. [2]

**Solution:** Yes. This follows directly from the definition of subsumption.

**(b)** Does T2 necessarily satisfy C1? Explain. [2]

**Solution:** No. There is no reason to expect test requirements generated by C1 to be satisfied by T2.

**(c)** If P contains a fault, and T2 reveals the fault, does T1 necessarily also reveal the fault? Explain. [2]

**Solution:** No. there is no requirement that test set T2 be a subset of test set T1. So, it could happen that T2 contains that one test that reveals the fault, and T1 doesn't.

**(d)** Explain why if a test set has covered every branch in a program, then the test set is guaranteed to also have covered every statement. [2]

**Solution:** Branch coverage subsumes statement coverage

**3.** Consider the `equals` method that compares two instances of `Person` in Listing 1:

```
public abstract class Person {
  private String firstName;
  private String lastName;

  public boolean equals(Person other){
    return this.firstName.equals(other.firstName) &&
       this.lastName.equals(other.lastName);
  }
}
```

Listing 1: Person.java

**(a)** Refactor the `equals` method to satisfy the Liskov Substitution Principle (LSP). [2]

**Solution:**

```
public abstract class Person {
  private String firstName;
  private String lastName;

  public boolean equals(Object other){
    return this.firstName.equals(other.firstName) &&
       this.lastName.equals(other.lastName);
  }
}
```

☐

**(b)** Use the contract specification of a method to explain how a derived class can be substitutable for its base class to satisfy LSP. [5]

**Solution:** The contract for a method can be specified in terms of its preconditions, postconditions and invariants. A derived class is substitutable for its base class if all the following is satisfied:

- Its preconditions are not stronger than the base class method
- Its postconditions are no weaker than the base class method.
- The invariants remain the same

**(c)** Explain why the program code in Listing 2 fails to satisfy LSP. [5]

```
public class Employee {
  private Employee manager;
  private double salary;

  public void assignManager(Employee manager) {
    this.manager = manager;
  }
  public void setSalary(double salary) {
    this.salary = salary;
  }
  public void calculateSalary(int rank) {
    setSalary(rank*10);
  }
}

class Manager extends Employee {
  @Override
  public void calculateSalary(int rank) {
    if(rank >10){
      return;
    }
    setSalary(rank*20);
  }
}
```

Listing 2: Employee.java and Manager.java

**Solution:** `calculateSalary` in Manager fails LSP because its preconditions are stronger than the base class method. This can refactored to satisfy LSP by removing the if conditional statement.

**(d)** State the Open-Closed principle of software design. [2]

**Solution:** Classes should be open for extension but closed for modification.

**(e)** The Lapsed listener problem often occurs in software implemented using the observer design pattern. State two consequences of this problem on the software. [2]

**Solution:**

1. Results in the leakage of the subscriber object.
2. Performance degradation due to the publisher sending redundant notifications to 'non existing' subscribers.