S211C/211C

DUBLIN INSTITUTE OF TECHNOLOGY

BSc. (Honours) Degree in Computer Science (Infrastructure)

Year 2

SUMMER EXAMINATIONS 2016

OBJECT ORIENTED PROGRAMMING [CMPU2016]

DR. SUSAN MCKEEVER & MARK FOLEY
DR. DEIRDRE LILLIS
MR. ALAN FAHEY

WEDNESDAY 11TH MAY 9.30 A.M. – 12.30 P.M

DURATION
3 HOURS

INSTRUCTIONS TO CANDIDATES

THERE ARE 2 SECTIONS ON THE PAPER: SECTION A AND SECTION B.
CANDIDATES MUST ANSWER **TWO** QUESTIONS OUT OF **EACH** SECTION
ANSWER FOUR QUESTIONS IN TOTAL.

ALL QUESTIONS CARRY EQUAL MARKS.

SECTION A
OO PROGRAMMING THROUGH PYTHON

ANSWER ANY TWO QUESTIONS OF THE THREE QUESTIONS IN THIS SECTION

Q1. The following code defines the start of a class to represent bank accounts:

```
class BankAccount(object):

    interest_rate = 0.3

    def __init__(self, name, number, balance):

       self.name = name

       self.number = number

       self.balance = balance

       return
```

(a) Name the class variables and the instance variables in the given code.

(5 marks)

(b) Add instance methods called deposit() and withdraw() which increase and decrease the balance of the account. Make sure the withdraw() method doesn't allow the account to go into overdraft. Add a third method called add_interest()which adds interest to the balance (the interest should be the interest rate multiplied by the current balance).

(10 marks)

(c) Create a subclass of BankAccount called StudentAccount. Every StudentAccount should have an overdraft limit of 1000. Write a constructor for the new class. Override the withdraw() method to make sure that students can withdraw money up to their overdraft limits.

(10 marks)

Q2 Consider the following program.

```
class Animal(object):

   population = 0

   def __init__(self, name):

      self.name = name

   def __str__(self):

      return "I am an instance of {}. My name is {}."
      .format(self.__class__, self.name)

   def __repr__(self):

      return self.__str__()

   def make_sound(self):

      return "{} is trying to speak but its method doesn’t do
much".format(self.name)


class Dog(Animal):

   def __init__(self, name, breed):

      super().__init__(name)

      self.breed = breed

   def __str__(self):

      print(super().__str__())

      return "My breed is {}".format(self.breed)

   def make_sound(self):

      return "{} says woof!".format(self.name)

class Cat(Animal):
   pass

animals = {'Felix': ('Cat', None), 'Fido': ('Dog', 'mutt'),
'Charlie': ('Dog', 'spaniel')}

animals_list = []

for k in animals:

   if animals[k][1]:

      animals_list.append(globals()[animals[k][0]](k,
animals[k][1]))

   else:

      animals_list.append(globals()[animals[k][0]](k))

   Animal.population+=1

for animal in animals_list:

   print(animal)

   print(animal.make_sound())

print("Animal population is {}".format(Animal.population))
```

(a) What output will the program produce? Explain the result.

(10 marks)

(b) What is the significance of the use of `globals()` in the program?

(5 marks)

(c) What is the significance of the use of `super()` in the program?

(5 marks)

(d) The `Cat` class appears to do nothing. Explain how a `Cat` instance would be created.

(5 marks)

Q3 Consider the following program which takes biopsy data relating to cancer patients and creates a classifier to predict whether any given set of test results for a patient indicates the presence or absence of a tumour.

T*his program is given to you for* ***reference****. You should refer to it but you don't have to understand all of it to answer the question. The use of comments in the code has been deliberately kept sparse.*

```
"""Classifier for sample breast cancer dataset
Process overview
1. Create training set from data
2. Create classifier using training dataset
3. Create test dataset
4. Use classifier to classify data in test set

Each data row consists of a patient id followed by nine indicators
followed by an overall result. Sample data row:
'1000025','5','1','1','1','2','1','3','1','1','2'.
In this case '1000025' is the patient id and the overall result is
indicated as '2' - malignant or '4' - benign.
"""

DATA_URL =
"http://archive.ics.uci.edu/ml/machine-learning-databases/breast-
cancer-wisconsin/breast-cancer-wisconsin.data"
PERCENT = 75

import httplib2

def create_data(DATA_URL):
    ts_list = []

    try:
        h = httplib2.Http(".cache")
        headers, fh = h.request(DATA_URL)
        fh = fh.decode().split("\n")

        row_count = 0
        for row in fh:
            try:
                row = row.strip()
                row_list = row.split(",")
                for i in range(1, len(row_list)-1):
                    row_list[i] = int(row_list[i])

                if row_list[-1] == '2':
                    row_list[-1] = 'm'
                elif row_list[-1] == '4':
                    row_list[-1] = 'b'
                else:
                    row_list[-1] = ''

                ts_list.append(tuple(row_list))

            except ValueError as v:
                print(row_list[0], v)
                continue

    except IOError as e:
        print(e)
    except ValueError as v:
        print(v)

    print(ts_list)
    return ts_list

def create_classifier(training_list):
    benign_attrs = [0]*9
    malignant_attrs = [0]*9
    benign_count = 0
    malignant_count = 0
    classifier_list = [0]*9

    # Compute the totals
    for record in training_list:
        if record[-1] == 'b':
            benign_count += 1
            for attribute in range(len(record[1:-1])):
                benign_attrs[attribute] += record[attribute+1]
        elif record[-1] == 'm':
            malignant_count += 1
            for attribute in range(len(record[1:-1])):
                malignant_attrs[attribute] += record[attribute+1]

    # Compute the averages
    for attribute in range(len(benign_attrs)):
        benign_attrs[attribute] = benign_attrs[attribute] /
benign_count
    for attribute in range(len(malignant_attrs)):
        malignant_attrs[attribute] = malignant_attrs[attribute] /
malignant_count

    # Compute the midpoints
    for attribute in range(len(classifier_list)):
        classifier_list[attribute] = (benign_attrs[attribute] +
malignant_attrs[attribute]) / 2

    print(classifier_list)
    return classifier_list

def create_test(test_list, classifier_list):
    false_count = 0
    true_count = 0
    total_count = 0

    temp_result_list = ['']*11
    for record in test_list:
        temp_result_list[0] = record[0]
        for attribute in range(len(record[1:-1])):
            if record[attribute+1] < classifier_list[attribute]:
                temp_result_list[attribute+1] = 'm'
            else:
                temp_result_list[attribute+1] = 'b'

        if temp_result_list.count('m') >= 5:
            temp_result_list[-1] = 'm'
        else:
            temp_result_list[-1] = 'b'

        print(temp_result_list, end=' ')
        total_count += 1
        if record[-1] == temp_result_list[-1]:
            true_count += 1
            print("CORRECT")
        else:
            false_count += 1
            print("FALSE")

    print("\nCORRECT: {}, {:.2%},  INCORRECT: {}, {:.2%},  TOTAL 
COUNT: {}"
         .format(true_count,true_count/total_count,false_count,
                 false_count/total_count,total_count))

def main():
    # Make a list of tuples from the raw data
    data_list = create_data(DATA_URL)

    # Break out our dataset into a training and test sets.
    training_list = data_list[:int(len(data_list)*PERCENT/100)]
    test_list = data_list[int(len(data_list)*PERCENT/100):]

    # Create the classifier values
    classifier_list = create_classifier(training_list)

    # Apply classifier against test file.
    create_test(test_list, classifier_list)

if __name__ == "__main__":
    main()
```

(a) Explain, in general terms, how a classifier such as this one works.

(15 marks)

(b) How does the use of functions in the program facilitate a divide-and-conquer strategy?

(5 marks)

(c) Why do we separate the training and test data sets? Explain the notion of "training" in this context.

(5 marks)

(d) How does the program handle rows containing bad data?

(3 marks)

(e) What does the following code in the create test function do?

```
if temp_result_list.count(’m’) >= 5:

   temp_result_list[-1] = ’m’

else:

   temp_result_list[-1] = ’b’
```

(2 marks)

SECTION B
OO PROGRAMMING THROUGH JAVA

ANSWER ANY TWO QUESTIONS OF THE THREE QUESTIONS IN THIS SECTION

Q1. (a) Figure 1 Section B shows a Unified Modelling Language (UML) hierarchy, showing a class hierarchy and interface. *Write the Java code* for each class/ interface shown. In addition, the following information is available:

1. An Employee's pay is calculated as salary multiplied by 12.
2. A PartimeEmployee's pay is calculated as salary multiplied by weekly hours. Pay for a PartimeEmployee can also be calculated by first checking if the part time employee is active before calculating the pay. If it is inactive, pay is set to zero.

If you cannot remember the exact Java syntax, write as close as an approximation as you can.

(15 marks)

(b) Pick out and explain examples of the following from the UML hierarchy

- Method overriding (2 marks)
- Method overloading (2 marks)
- Encapsulation (2 marks)
- Inheritance (2 marks)
- Method signatures (2 marks)

(10 marks)

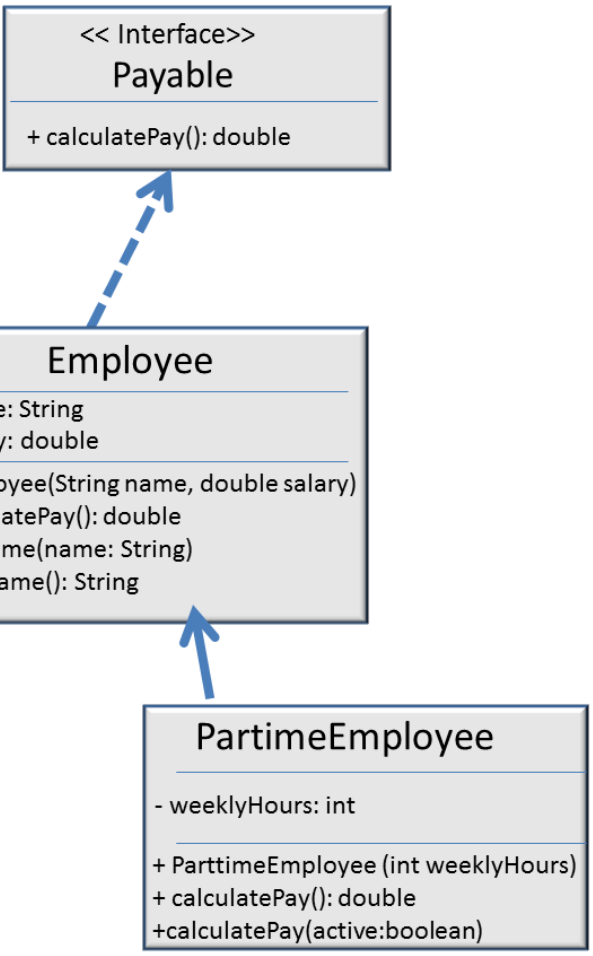

Figure 1 Section B: Employee hierarchy

Q2. (a) Explain in your own words how *event programming* is implemented for a graphical user interface in Java. Use an example of a clickable button (JButton). Make sure to include each of the implementation steps required.

(12 marks)

(b) Figure 2 Section B Sample Java class shows the Java code for an Account class. Identify and explain the following

- Four compile errors that will prevent the code from compiling (4 marks)
- Two specific coding standard improvements (2 marks)

(6 marks)

```
public class account implements ValidatedAccount{
      public String Accountname;
      public double AcctBalance;
      public String sortCode;

      Account (String Accountname, int AcctBalance,
                                  String sortCode)  {
            this.Accountname = Accountame;
            this.acctBalance = acctBalance;
            this.sortCode    = sortCode;
      }

public double Getbalance (amount){
   AcctBalance = AcctBalance - amount;

}
```

Figure 2 Section B: Sample Java class

(c) A Java developer wishes to create an ArrayList of String objects and has noticed that the ArrayList class in the Java API begins as:

```
Public class ArrayList<E>
```

Explain the <E> notation used and how *Generics* are useful in the creation of lists in Java.

(7 marks)

Q3 (a) Explain the *differences* between the `Array` and `ArrayList` classes in Java. Use an example to support your answer.

(5 marks)

(b) Explain the purpose of the following two interfaces that are supplied in the Java API. In your answer, explain clearly how these interfaces are useful in helping Java developers to avoid re-writing code (i.e. “reinventing the wheel”):

- `Collection` (5 marks)
- `Iterator` (5 marks)

(10 marks)

(c) Explain clearly the differences between each of the following three access modifiers in Java:

- `public` (2 marks)
- `protected` (3 marks)
- `private` (2 marks)

(7 marks)

(d) Explain the meaning of the `implements` keyword in Java.

(3 marks)