**CS1180**

**THE UNIVERSITY OF WARWICK**

**First Year Examinations: April 2016**

**Programming for Computer Scientists**

---

**Time allowed: 2 hours**

Answer **ALL** the questions. All questions carry 20 marks each.

Most answers require a single statement or a short piece of Java code to be written. Do not give complete programs or supply irrelevant declarations or input-output statements unless explicitly asked for. Number your answers clearly.

Read carefully the instructions on the answer book and make sure that the particulars required are entered on each answer book.

---



Continued

1. **Primitive Data Types and Control Statements**

   (a) Java has four primitive integer variable data types. The smallest of which is a `byte` (8 bits). Give the binary and decimal representation of both the largest and the smallest value a `byte` can hold. [2]

   (b) Floating point numbers in Java are represented using IEEE-754 notation. Given an 8-bit floating point number, where the exponent is 3 bits and the fraction is 4 bits, and the number is calculated using the formula:

   $$(\text{sign})1.(\text{fraction}) \times 2^{(\text{exponent}-3)}$$

   calculate the value of the number $01101100$. [5]

   (c) The following code will not compile. Explain why this is the case and give two valid ways to resolve the error. [3]

```
public static void main(String args[]) {
    double a = 8374;
    int b = a;
    System.out.println(b);
}
```

   (d) Program control in Java can be managed using `if` and `switch` statements. Rewrite the following code in the most concise form possible using an `if-else` statement. [5]

```
switch (a) {
    case 4: System.out.println("Option 4");
    case 2: System.out.println("Option 2");
            break;
    case 1: System.out.println("Option 1");
    default: System.out.println("Default");
}
```

   (e) Given `x = 0`, determine the validity and the truth value of the following statements, as well as the value of `x` following evaluation.

   i. `(x == 0) || (x++ < 2)`

   ii. `(x > 2) & (x++ < 2)`

   iii. `(x != 0) & ((100 / x) != 2)`

   [5]



Continued

---

2. **Iterative Statements, Methods, Arrays and Recursion**

   (a) Using iterative statements, write code to calculate the result of the following equation.

$$\frac{\sum_{x=1}^{100}(x^2+3)}{2}$$

   [5]

   (b) Write a method that, when given an integer value, checks whether the value is a prime number or not. [5]

   (c) Using your previous answer, write code that fills an array with the first 20 prime numbers. [6]

   (d) Write a recursive method that takes an array of integers (named `a`) as an input parameter and finds the value of the largest element. [4]

---

3. **Object Oriented Programming**

   (a) It is said that the "four pillars" of object oriented programming are: Abstraction, Encapsulation, Inheritance and Polymorphism. Explain the meaning of **two** of these concepts with relation to OO programming in Java. [4]

   (b) Fractional values in Java are usually stored using the IEEE-754 floating point standard. Inherent in the standard is a possible loss of precision; this can be avoided for *common* fractions by storing the value as a numerator and a denominator separately.

      i. Design a class, `Fraction`, that stores a common fraction as an integer numerator and an integer denominator. You should provide three constructors (a default constructor, an $\frac{n}{1}$ constructor and an $\frac{n}{m}$ constructor). [3]

      ii. Write appropriate accessor and mutator methods for your class. [2]

      iii. Write appropriate methods for adding and subtracting two `Fraction` objects. [3]

      iv. Write appropriate methods for multiplication and division of two `Fraction` objects. [3]

      v. Write a method, `reduce`, that returns the Fraction in its most reduced form. [5]



Continued

4. **Inheritance, Abstract Classes and Interfaces**

   (a) You are working for a company developing an employment database application. The database will store information about each employee (such as their name, date of birth, contact information, etc.) and will also provide access management for the company's premises.

      i. Design an abstract `Employee` class to store the following attributes: forename, surname, date of birth and annual salary. [3]

      ii. Add methods to your `Employee` class to modify the attributes and additionally add an abstract method to add overtime to an employee's monthly wage. [3]

      iii. Programmers earn overtime at $\frac{1}{1500}$th of their annual salary per hour. Design a concrete `Programmer` class with an additional variable to hold monthly overtime payments (you may assume that this counter is reset automatically). [3]

      iv. A salesperson earns overtime at $\frac{1}{1000}$th of their annual salary per hour. Additionally they are contactable via a personal office phone number. Design a concrete `Salesperson` class that holds this additional information and includes a method to retrieve their external and internal contact numbers (where the internal number is the final 5 digits of their 11 digit external phone number). [4]

   (b) The access management in the building is controlled by querying an `Employee` object. There are two levels of access in the building: BuildingAccess and SecureAccess. Employees with BuildingAccess should be expected to have a method to retrieve the employee ID; employees with SecureAccess should have a method for verifying a PIN code.

      i. Design two interfaces (`BuildingAccess` and `SecureAccess`) that can be used to provide this access management. [2]

      ii. Within the company, programmers can access the building and secure resources; salespeople are granted building access only. Show how the class declarations provided previously would be amended to account for this. [2]

      iii. Finally, given the following method:

```
public boolean AccessBuilding(Employee e) {
    // does employee have required level of access
        addLog(((BuildingAccess) e).getEmployeeID());
        return true;
    // else
        return false;
}
```

      Demonstrate how this method would check if the provided `Employee` has the required level of access for the building. Additionally, write the method `AccessSecure` to check access to secure resources. [3]

5. **Exceptions and Generics**

(a) In Java, erroneous behaviour often results in an Exception being generated. An Exception allows a programmer an opportunity to handle an error gracefully and possibly take corrective measures.

i. Performing a division by zero in Java causes an exception when dealing with integer division. However, for floating point numbers, the special value "`Inf`" is returned and no exception is raised. Write a *checked* exception class that will be raised when a floating point division by zero is about to occur. [3]

ii. Write a method that divides one floating point number by another and returns the result. In the event that the denominator is zero, an exception should be generated. [4]

iii. With a code example, demonstrate how you would use your division method in a Java application. [3]

(b) In Java, *Generics* can be used to provide type information at compile-time.

i. Explain two of the motivating factors behind using Generics in Java. [2]

ii. Design a `KeyValuePair` class where the value **must** be a numerical value. [6]

iii. Show how you would initialise a `KeyValuePair` object in Java. [2]



End