PROG2004
Object Oriented Programming

Summary

Title	**Assessment 1**
Type	Programming
Due Date	**Monday 15 July 11:59 pm (Start of Week 3)**
Length	15 hours
Weighting	20%
Academic Integrity	Contract cheating and the use of GenAI, such as ChatGPT, in this assignment are **strictly prohibited**. **Any breach may have severe consequences.**
Submission	You will need to submit the following to the submission link provided for this assignment on the MySCU site: • The JAVA project with all your source code files. • The link to your GitHub repository. • A video explaining your code.
Unit Learning Outcomes	This assessment task maps to the following ULOs: • ULO1: explain, compare, and apply advanced class design concepts • ULO2: apply object-oriented programming principles to solve intermediate problems

Rationale

The purpose of this assessment is to test your ability to:

- ULO1: Explain, compare, and apply advanced class design concepts; and
- ULO2: Apply object-oriented programming principles to solve intermediate problems

Your work will demonstrate your learning over the first two modules of this unit.

Task Description

In this assignment, your task is to write JAVA codes that manage **a simple appointment system for a health service**. A patient may make an appointment with different types of health professionals. Health professional is a common term used to describe health workers (e.g., general practitioners, specialists, surgeons, dietitians, nurse, etc).

To limit the complexity, this assessment **focuses on two types of health professionals** and **the booking is only for the current day**. Your solution **must apply the inheritance concept** and utilise **a suitable collection** to handle the bookings.

In this assessment, you MUST:

- Use **IntelliJ** and **JAVA language** to develop the solution.
- Submit **JAVA project** with all your source code files to MySCU link.
- **Demonstrate your work progress** by regularly uploading your code to your GitHub repository over the last two weeks.
- **Record a video** to explain your code to demonstrate your understanding.

If you fail to do any of the above, you will lose marks and be required to attend an interview to explain your code. If you cannot explain your code, you will be submitted for academic misconduct.

Getting Started

To get started:

- Create a new Java project called **username-A1** in IntelliJ.
- In the **src** directory, create a new class called **AssignmentOne**.
- In the AssignmentOne class, create the **main method**.

For this assessment, find a website of any health service near your place (e.g., Southport Method Medical Centre - https://southportmetromedicalcentre.com.au) to explore all the different types of health professionals with that patients can make appointments. While you are doing this, have a look at the different health professional types as they go from more general to more specific. At the top level of the Inheritance hierarchy, you need a generic health professional.

Note that this assessment does not endorse any health services. The reference was only used to give students an understanding of the case study.

Part 1 – Base class

This part is to create a base class that will be extended to create different types of health professionals.

In your Java project, create a class named **HealthProfessional**.

The class must have the following at a minimum:

- Required instance variables: ID (numbers only) and name.
- Another instance variable that is relevant to describe the basic information regardless of the doctor type.
- A default constructor.
- A second constructor that initialises all the instance variables.
- A method that prints all instance variables.

Part 2 – Child classes

This part is to create two child classes representing different types of health professional. To limit the complexity, this assessment **focuses on two types: general practitioner and any other type that you prefer.** Therefore, in the inheritance hierarchy, under class HealthProfessional, you would need to handle general practitioners and specialists.

In your Java project, create **one child class** named **GeneralPractitioner** that extends the base class.

The child class must have the following at a minimum:

- At least another one instance variable that is relevant and suitable to differentiate between general practitioner and another health professional type.
- A default constructor
- A second constructor that initialises all the instance variables (including the instance variables in the base class)

- A method that prints the health professional details, **including the health professional type** (if it is a general practitioner or other health professional type). E.g., "The doctor details are:" followed by all instance variables formatted for readability (including the instance variables in the base class)
- Any other methods or variables that you feel are necessary for the class to be usable in the program

Now, **think of another type of health professional** that you want to create that can be extended from HealthProfessional class. In your Java project, create **another one child class** and give any class name that extends the base class.

Similar to **GeneralPractitioner**, this new child class must have the following at a minimum:

- At least another one instance variable that is relevant and suitable to differentiate between this class and general practitioner.
- A default constructor
- A second constructor that initialises all the instance variables (including the instance variables in the base class)
- A method that prints the health professional details, **including the health professional type** (if it is a general practitioner or dietitian). E.g., "The doctor details are:" followed by all instance variables formatted for readability (including the instance variables in the base class)
- Any other methods or variables that you feel are necessary for the class to be usable in the program

Part 3 – Using classes and objects

This part is to use the classes created above and create objects of different types of health professionals. You are not required to use a collection to store the general practitioners and dietitieans but you may use it if you want to.

In the **main method**:

- Add the following comment - // Part 3 – Using classes and objects
- Write the **code to create three objects of General Practitioners** and **two objects of another health professional type**.
- Create a comment to mention where you get the doctor's information.
- Print all health professionals you created (including the information from the base class). **Use the method** you just created to demonstrate your understanding
- Add the following code - System.out.println("----------");

Part 4 – Class Appointment

This part is to create a class to accommodate an appointment made by a patient. When a patient wants to make an appointment, we need to store basic patient's detail, preferred time slot, and which doctor the patient wants to meet.

In your Java project, create a new class named named **Appointment**. The class must have the following at a minimum:

- Instance variables for patient details: name and mobile phone. You are not required to create a Patient class, but you may create it if you want to.
- Instance variable for the preferred time slot (e.g., 08:00, 10:00, 14:30).
- The selected doctor (general practitioner or another health professional type). This should be an object of the child class.
- A default constructor.
- A second constructor that initialises all the instance variables.
- A method that prints all instance variables.

Part 5 – Collection of Appointments

This part is to create a collection of appointments using **ArrayList** and demonstrate how the appointments work. We may add a new appointment, print existing appointments and cancel an appointment.

In the **main method**, write the code to:

- Add the following comment - // Part 5 – Collection of appointments
- **Declare an ArrayList** that can store instances (objects) of Appointment class.
- Create a method named **CreateAppointment** to create a new booking and add it to the ArrayList. Since inheritance is applied, the method should be able to handle if the appointment is to meet any different health professional types (think carefully). Also, make sure all required information is supplied when creating a new appointment. Otherwise, the appointment can not be created.
- Create a method named **PrintExistingAppointment** to display existing appointments in the array list. If no existing appointments, print a message to indicate this.
- Create a method named **CancelBooking** to cancel a booking by a patient's mobile phone. If the mobile phone is not found in the existing appointment, print a message to indicate this error.
- Add the following code - System.out.println("------------------------------");

Lastly, demonstrate the collection and methods created aboved by doing these:

- Make 2 appointments with general practitioners.
- Make another 2 appintment with another health professional types.

- Print existing appointments.
- Cancel one of the existing appointments
- Print again existing appointments to demonstrate the updated collection of appointments.

Your main JAVA method should have the following information:

```
// Part 3 - Using classes and objects
Code demonstrating part 3
System.out.println("------------------------------");
// Part 5 - Collection of appointments
Code demonstrating part 5
System.out.println("------------------------------");
```

NOTE: Make sure that none of the code demonstrating each part of the assignment is commented out. Your marker will run your main method and will be expecting to see the output for all parts of the assignment. If the output for any part is missing, you WILL lose marks for that part. You can comment it out when you are developing; however, when you submit your assignment, the main method must contain all the code required for all parts of the assignment.

Use GitHub

You must **create a repository on GitHub** to store your project work with all files and documents. You **must show your work progress** in this assignment by regularly committing your project to the GitHub repository. **Failing to show the correct work progress will fail the assignment.**

Create a video

In your video, ensure you explain these

- Explaining how you implement inheritance in Parts 1-3 .
- Explaining how you demonstrated polymorphism in Part 3.
- Explaining how you work with ArrayList and Class to handle the appointments.

Your video does not need to be long or go into a lot of detail. You should be able to do all the above in <= 5 minutes; however, the video must demonstrate that:

- You understand how to implement inheritance and collection to handle appointments for different types of health professionals.
- You understand the code you are submitting in the remaining parts, and did not use ChatGPT or similar GenAI to generate it.

Upload your video to your SCU OneDrive and create a sharable link to it.

Task Submission

You are required to submit the following items:

- The JAVA project with all your source code files (to MySCU submission link). **Zip your project into a file called username-A1.zip**
- The link to your GitHub repository (put the link on MySCU submission comment)
- A short video to explain your code part by part (upload the video file to submission if the size allows, or put the video link on MySCU submission comment).

Resources

To complete the task, you are recommended to:

- Study modules 1 and 2 materials and complete all learning activities
- Take an active role in the weekly tutorial and workshop.

Assessment Criteria

Please refer to the rubric provided in the assessment folder for the assessment criteria. Marking criteria include:

- Java code compiles with Java 17 LTS
- Use of correct coding style, including the use of comments
- Accuracy of coding
- Use of suitable coding structures
- Correct submission and naming conventions of assessment items as required

Academic Integrity

At Southern Cross University, academic integrity means behaving with the values of honesty, fairness, trustworthiness, courage, responsibility and respect in relation to academic work.

The Southern Cross University Academic Integrity Framework aims to develop a holistic, systematic and consistent approach to addressing academic integrity across the entire University. For more information, see: SCU Academic Integrity Framework

NOTE: **Academic Integrity breaches include** unacceptable use of generative artificial intelligence (GenAI) tools, the use of GenAI has not been appropriately acknowledged or is beyond the acceptable limit as defined in the assessment, poor referencing, not identifying direct quotations correctly, close paraphrasing, plagiarism, recycling, misrepresentation, collusion, cheating, contract cheating, fabricating information.

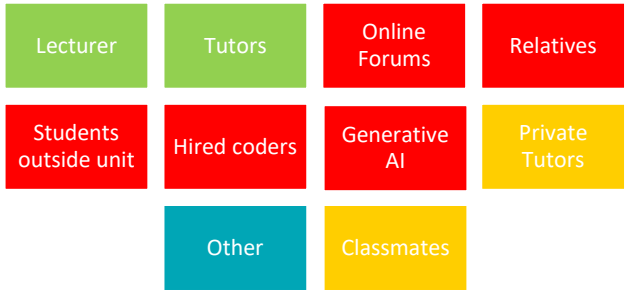
Use of Gen AI such as ChatGPT

Generative artificial intelligence (GenAI) tools, such as ChatGPT, **must not be used** for this assessment task. You are required to demonstrate that you have developed the unit's skills and knowledge without the support of GenAI. If you use GenAI tools in your assessment task, it may result in an academic integrity breach against you, as described in the Student Academic and Non-Academic Misconduct Rules, Section 3.

Please note that your assignment will be submitted to a similarity detection service by your marker. Advanced plagiarism detection software will be used that compares the answers for all students to all questions and flags any similarities in assessments. If your marker has any suspicion that you had help with your code or that your work is not your own, you will be asked to come to a meeting with your marker to explain your code. Any student who is unable to explain their code will be submitted for academic misconduct.

Getting Help

This diagram will help you understand where you can get help:



Encouraged **Attribution Required**
Not acceptable **Ask tutor**

Be aware that if you do get help from one of the red sources, you will be reported for academic misconduct, which may have serious penalties. Please visit the following link for the guidelines: **https://bit.ly/scuAcadMisconduct**

Special Consideration

Please refer to the Special Consideration section of Policy. https://policies.scu.edu.au/document/view-current.php?id=140

Late Submissions & Penalties

Please refer to the Late Submission & Penalties section of Policy. https://policies.scu.edu.au/view.current.php?id=00255

Grades & Feedback

Assessments that have been submitted by the due date will receive an SCU grade. Grades and feedback will be posted to the 'Grades and 'Feedback' section on the Blackboard unit site. Please allow 7 days for marks to be posted.

Description of SCU Grades

High Distinction:

The 'student's performance, in addition to satisfying all of the basic learning requirements, demonstrates distinctive insight and ability in researching, analysing and applying relevant skills and concepts, and shows exceptional ability to synthesise, integrate and evaluate knowledge. The 'student's performance could be described as outstanding in relation to the learning requirements specified.

Distinction:

The 'student's performance, in addition to satisfying all of the basic learning requirements, demonstrates distinctive insight and ability in researching, analysing and applying relevant skills and concepts, and shows a well-developed ability to synthesise, integrate and evaluate knowledge. The 'student's performance could be described as distinguished in relation to the learning requirements specified.

Credit:

The 'student's performance, in addition to satisfying all of the basic learning requirements specified, demonstrates insight and ability in researching, analysing and applying relevant skills and concepts. The 'student's performance could be described as competent in relation to the learning requirements specified.

Pass:

The 'student's performance satisfies all of the basic learning requirements specified and provides a sound basis for proceeding to higher-level studies in the subject area. The 'student's performance could be described as satisfactory in relation to the learning requirements specified.

Fail:

The 'student's performance fails to satisfy the learning requirements specified.