PROG2004 OBJECT ORIENTED PROGRAMMING

Summary

Title	Assessment 1 – Programming tasks - a simple appointment system for a health service
Type	Programming
Due Date	Monday 11 November 2024 11:59 pm AEST (Start of Week 3)
Length	NA
Weighting	20%
Academic Integrity (See below for limits of use where GenAI is permitted)	You **may use** GenAI tools to get some ideas or insight about particular problems in code. However, you **MUST NOT** generate a complete solution code. Please refer to the **Academic Integrity section** below to see what you can and cannot do with the GenAI tools.
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 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

This assessment aims to assess students' ability to understand and apply object-oriented programming concepts and principles to solve intermediate problems. It addresses unit learning outcomes 1 and 2 and relates to modules 1 and 2.

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, nurses, 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 **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 and demonstrate your understanding.

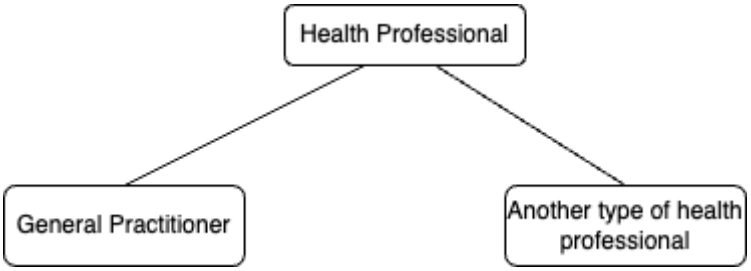
If you fail to do any of the above, you will lose marks and be required to attend an interview to explain your code. If you cannot explain your code, you will be submitted for academic misconduct.

Task Instructions

To get started:

- Create a new Java project called **username-A1**.
- 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 professionals. 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 the other type of health professional.

In your Java project, create **one child class** named **GeneralPractitioner** that extends the base class and **another child class for the other type of health professional** that also extends the base class.

The child class must have the following at a minimum:

- At least another instance variable that is relevant and suitable to differentiate between a 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 health professional 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 uses the classes created above to create objects for different types of health professionals. You are not required to use a collection to store the general practitioners and the other type of health professionals, 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 the other health professional type**.
- **Use the methods** you created in the base and child classes to print the details of all the health professionals you created in the above step (including the information from the base class).
- 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 details, preferred time slot, and which doctor the patient wants to meet.

In your Java project, create a new class 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 **printExistingAppointments** to display existing appointments in the **ArrayList**. If there are no existing appointments, print a message to indicate this.
- Create a method named **cancelBooking** to cancel a booking using 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 above by doing these:

- Make 2 appointments with general practitioners.
- Make another 2 appointments with the other type of health professional.
- 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. In each commit, you need to provide a clear explanation of what changes you have made in this commit. **Failing to show the correct work progress will fail the assignment.**

Create a video

In your video, ensure you explain:

- how you implemented inheritance in Parts 1-3,
- how you demonstrated polymorphism in Part 3, and
- how you worked with ArrayList and Class to handle the appointments.

Your video does not need to be long or go into much detail. You should be able to do all the above in <= 5 minutes; however, the video must demonstrate that you understand the code you are submitting and you did not use ChatGPT or a similar GenAI tool to generate it.

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

Resources

Use the following resources to support you when working on this assessment.

- Study modules 1 and 2 materials and complete all learning activities.
- Take an active role in the weekly tutorial and workshop.
- Java API documentation https://docs.oracle.com/en/java/javase/22/docs/api/index.html

Referencing Style Resource

NA

Task Submission

You are required to submit the following items:

- The JAVA project with all your source code files. **Zip your project into a file called username-A1.zip and upload the file.**
- The link to your GitHub repository. **Add the link in the comments.**
- The link to your short video explaining your code part by part. **Add the link in the comments.**

Resubmit policy: This assessment is not eligible for a re-submit.

Assessment Criteria

Please refer to the marking rubric for more details. Marking criteria include:

- Java code compiles
- 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.

At SCU the use of GenAI tools is acceptable, *unless it is beyond the acceptable limit as defined in the Assessment Item by the Unit Assessor*.

GenAI May be Used
Generative artificial intelligence (GenAI) tools, such as ChatGPT, **may be used** for this assessment task. If you use GenAI tools, you must use these ethically and acknowledge their use. To find out how to reference GenAI in your work, consult the referencing style for your unit via the Library referencing guides. If you are not sure how to, or how much, you can use GenAI tools in your studies, contact your Unit Assessor. If you use GenAI tools without acknowledgment, it may result in an academic integrity breach against you, as described in the Student Academic and Non-Academic Misconduct Rules, Section 3.

You **may use** Generative Artificial Intelligence (GenAI) tools, such as ChatGPT or Copilot, for this assessment task **to get some ideas or insight about particular problems in code**. It is similar when you try to find a snippet of code for doing a particular task in Stack Overflow. For example, **you must make your own effort to modify, refine, or improve the code to solve the assessment problems**. Think of it as a tool – a quick way to access information – or a (free) tutor to answer your questions. However, just as if you Googled something, you still need to evaluate the information to determine its accuracy and relevance. ***If you have used a GenAI tool in this assessment, you must document how you used it and how it assisted you in completing your assessment tasks***. ***Failing to do that will be subject to an academic integrity investigation***.

You **cannot use AI to generate a complete solution code**. You need to put your own effort into building the solution code for the assessments to demonstrate the required skills. Refer to assessment information in the Assessment Tasks and Submission area for details.

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.

… continued on next page ...

Assessment Rubric

Marking Criteria and % allocation	High Distinction (85–100%)	Distinction (75–84%)	Credit (65–74%)	Pass (50–64%)	Fail 0–49%
Apply object-oriented programming principles and develop advanced class design to solve health professional scenario (Parts 1-2) **(ULOs 1-2)** **25%**	Demonstrates **exceptional** understanding and application of object-oriented principles (abstraction, encapsulation, inheritance, polymorphism, class) to solve complex health professional problem. Designs classes that are **highly cohesive and effectively organised** using advanced class design.	Demonstrates a **solid understanding** and application of object-oriented principles **with minor errors or omissions** in solving health professional problems. Designs classes that **generally adhere** to principles of cohesion and advanced class design **with occasional errors or less effective** implementation.	Demonstrates a **basic understanding** of object-oriented principles but **with notable gaps or errors** in applying them to health professional problems. Designs classes that show **basic cohesion** but lack sophistication in advanced class design.	Demonstrates a **minimal understanding** of object-oriented principles with **substantial errors or misunderstandings** in applying them to health professional problems. Designs classes that are **poorly organised or inefficient**.	**Fails to demonstrate** a basic understanding of object-oriented principles, with critical errors or complete lack of application to health professional problems. Designs classes that are **fundamentally flawed** and **do not meet** basic requirements.
Implement advanced class concept to manage health professional data (Part 3) **(ULOs 1-2)** **10%**	Implements advanced class **effectively and efficiently without errors** to manage health professional data.	Implements advanced class **with some errors or oversights** in managing health professional data. creativity and innovation.	Implements advanced class with **significant errors or oversights** in managing health professional data.	Implements advanced class **with fundamental errors** or **lacks basic understanding** of managing health professional data.	**Fails to implement advanced class** effectively, **with critical errors** in managing health professional data or **no implementation**.

Apply object-oriented programming principles and develop advanced class design to solve appointment scenario (Part 4) **(ULOs 1-2)** **20%**	Demonstrates **exceptional** understanding and application of object-oriented principles (abstraction, encapsulation, inheritance, polymorphism, class) to solve complex appointment problem. Designs classes that are **highly cohesive and effectively organised** using advanced class design.	Demonstrates a **solid understanding** and application of object-oriented principles **with minor errors or omissions** in solving appointment problems. Designs classes that **generally adhere** to principles of cohesion and advanced class design **with occasional errors or less effective** implementation.	Demonstrates a **basic understanding** of object-oriented principles but **with notable gaps or errors** in applying them to appointment problems. Designs classes that show **basic cohesion** but lack sophistication in advanced class design.	Demonstrates a **minimal understanding** of object-oriented principles with **substantial errors or misunderstandings** in applying them to appointment problems. Designs classes that are **poorly organised or inefficient**.	**Fails to demonstrate** a basic understanding of object-oriented principles, with critical errors or complete lack of application to appointment problems. Designs classes that are **fundamentally flawed** and **do not meet** basic requirements.
Implement advanced class concept to manage appointment data (Part 5) **(ULOs 1-2)** **25%**	Implements advanced class **effectively and efficiently without errors** to manage health professional data.	Implements advanced class **with some errors or oversights** in managing health professional data. creativity and innovation.	Implements advanced class with **significant errors or oversights** in managing health professional data.	Implements advanced class **with fundamental errors** or **lacks basic understanding** of managing health professional data.	**Fails to implement advanced class** effectively, **with critical errors** in managing health professional data or **no implementation**.
Accuracy, efficiency, validations and compatibility **(ULOs 1-2)**	Demonstrates **exceptional accuracy, efficiency, validations** to serve the objectives and requirements.	Demonstrates **a solid accuracy, efficiency, and validations** with minor errors or omissions in serving	Demonstrates a **basic accuracy, efficiency, and validations** with **notable gaps of errors** in serving the	Demonstrates a **basic accuracy, efficiency, and validations** with **notable gaps of errors** in serving the	**Fails to demonstrate** a basic accuracy, efficiency, and validations to serve the objectives and requirements.

10%	JAVA code can be **compiled and run without any issues.**	the objectives and requirements. JAVA code can be **compiled and run without any issues.**	objectives and requirements. JAVA code is **compiled** and can be run **with some minor issues.**	objectives and requirements. JAVA code is **compiled** and can be run **with major issues.**	JAVA code is not compiled and has significant errors/fails to be run.
Concept understanding (Comment, GitHub, video) **(ULOs 1-2)** **10%**	Demonstrates a **profound understanding** of object-oriented programming principles and advanced class design for the case study **through code comments, work progress in GitHub, and video**.	Demonstrates a **thorough understanding** of object-oriented programming principles and advanced class design for the case study **through code comments, work progress in GitHub, and video.**	Demonstrates a **basic understanding** of object-oriented programming principles and advanced class design for the case study **through code comments, work progress in GitHub, and video.**	Demonstrates a **minimal understanding** of object-oriented programming principles and advanced class design for the case study **through code comments, work progress in GitHub, and video.**	**Fails to demonstrate** a basic understanding of object-oriented programming principles and advanced class design for the case study **through code comments, work progress in GitHub, and video.**

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.