Applied session (Bootcamp)

Applied session objectives - Bootcamp Week 2

In this Bootcamp, you will:

- design and implement a multi-class program
- use fields (a.k.a. attributes) in Java classes
- use methods (a.k.a. operations) in Java classes
- use git to store your code and documentation.

Bootcamp specifications

Assessment: This applied session is worth 1% of your final mark for FIT2099. It will be awarded in the form of a checkpoint to be completed during the applied session next week as it directly contributes to the bootcamp's final delivery in Week 6.

Please do note the due date for every bootcamp is 11:59 PM (midnight) on the day before your allocated applied class, so please be mindful of that. For example, if your applied session is on Wednesday, then your week 2 bootcamp will be due on Tuesday of week 3 at 11:59 PM. Please also note that class attendance is **MANDATORY** to get your checkpoint / interview done, unless you have a special consideration granted.

Welcome!

Welcome to the first bootcamp! We will create a **Camp Simulator**, a console-based Java application, in the next 4 weeks (Bootcamps 2–5; you have completed Bootcamp 1). We will start small (e.g., only items that we have inside backpack) and gradually add complex actionable features (e.g., interacting with items such as ignite a lantern, drink from a bottle, and more). This bootcamp project will be a great opportunity to get your head around the foundations of Java and learn essential implementations of Object-oriented design principles in a fun and engaging way!

Please, read all the requirements listed below first, then start working on the design before doing any coding.

Common Requirements (REQ 1 & 2)

There will be 2 common requirements for each bootcamp and they are reflected in the marking rubric accordingly. These 2 requirements will remain the same and consistent across all bootcamps and will be assessed every time. Please read them very carefully to get yourself familiar with the common requirements and be ready for future bootcamps as well!

Link: https://edstem.org/au/courses/25302/lessons/84428/slides/576351

Specific Tasks (REQ 3)

There will be 4 tasks under REQ3 for each bootcamp. These are designated to help you consolidate and assess your understanding of that specific week's content and are aligned with the Applied session objectives. The 4 tasks will be different for each week and will be built on top of the previous week's bootcamp. These tasks will cover both coding and design skills we anticipate you to hold throughout the learning journey.

Link: https://edstem.org/au/courses/25302/lessons/84428/slides/576352

Remember, if you encounter any issues or have any questions, please don't hesitate to reach out to a TA for

help! If you still have questions after the applied session, you can post them on the Ed discussion forum or attend a consultation.

Important Notes:
* Show your marker a system that runs as described. You must also provide a UML class diagram showing the classes and the relationships between them.
* Show your marker that your system follows good Java practices and meets non-functional requirements. This includes:
  - implementing the structure precisely as specified,
  - hiding implementation details as much as possible,
  - having well-chosen names for attributes and methods (other than those we have named for you), and
  - correctly distinguishing between associations and dependencies in the class diagram
* Applied Session TAs cannot mark students who are not in their Allocate+ scheduled applied session.
* You will not be able to complete your applied session work in a applied session, so please, attempt it before the applied session.
*** Showing the code to your marker is not enough to get the applied session mark. A student must attend and pass the applied session interview a week after the applied session.**
- This means not only you do need to prepare your solutions but you should prepare for the interview.
- You cannot get marks for presenting code you do not completely understand.

Bootcamp Common REQs (REQ 1 & 2)

Requirement 1. Using git

From now on, you will be required to keep your Bootcamp source code on **https://git.infotech.monash.edu**. By the end of this week, you should have been provided with a personal repository. If you don't know about this (for example, if you just enrolled), please ask your TA IMMEDIATELY so we can create one for you.

If you haven't successfully set up your git repository on your local machine (e.g., laptop, PC), please go to ***Ed Lessons → Week 1→ Supplementary Materials/Applied Session (Bootcamp)***. Your marker will assess your bootcamp using the latest version of the project in your remote Git repository (i.e., the latest "push" in Git history). Since this assessment is directly done via your Git repository, you will **NOT** need to submit anything to Moodle for any bootcamp activities (NOTE: Assignment submission will be slightly different, stay tuned!). Bootcamp work that is not pushed into your remote Git repository will most likely receive ***ZERO*** marks.

Please make sure all Git-related issues are resolved before attempting this bootcamp. Always ensure that you check existing materials or external resources (e.g., Stackoverflow, YouTube, etc.) to see if they can answer your question or solve the issue. We encourage you to do this self-exploration as it is part of learning that would benefit you in the long run. If you are still facing difficulties in solving such technical issue(s), please do not hesitate to consult your TA or attend any consultation session to get assistance.

Our bootcamps are structured to be continuous, meaning that each week you will be required to update your work according to the specific requirements or tasks assigned for that week's Bootcamp. In other words, you are developing just **ONE** project for this Bootcamp. The following sub-section provides guidance on effectively managing your weekly work. It offers valuable insights on how to stay organised and make the most of each week's Bootcamp.

Git Branch Practice

Please create a new branch `bootcamp-week-2` from your main branch. Ensure that you merge it back into the main branch before the interview/assessment next week. **We only assess work inside the `main` branch.**

If you didn't have time to work on creating a branch and merging in Week 1, don't worry! You can practice with git branches this week. Please go through the material on how to create a Git branch and/or ask your TA for a demonstration.

Although it's not a mandatory requirement to use git branches during the bootcamp, it is **highly recommended** you practice using it during the bootcamp as you will need to utilise it when conducting the

main assignments later in the semester (e.g., collaborate with other students as a team to develop a project).

Requirement 2. UML Class Diagram

Before we start coding in this week's Bootcamp session, let's spend some time creating a UML class diagram for the system we plan to work on (specific feature requirements are on the next slide - Requirement 3). This will help us understand the problem more clearly and make the project easier to scale and collaborate on. Remember that the system should address various requirement items, so be sure to read the entire lab sheet carefully. If you don't feel confident or stuck on a particular requirement, it may also be a good idea to do some coding while creating the UML diagram. This **iterative** process helps you simultaneously shape and sharpen design and implementation.

We'll be working on this system for the next few weeks, but for now, let's only include what's covered in this week's Bootcamp. We'll be updating the class diagram each week, so let's keep it simple and only show the classes and relationships between them, such as associations, dependencies and other relationships in UML. You DO NOT need to include attributes or methods in the UML class diagram.

Again, just to reiterate: you don't need to include attributes or methods in the diagram.

During today's applied session...

Sketch the class diagram and show it to a TA (Teaching Associate / Tutor) to get some early feedback. Even if you haven't finished all the requirement items by the end of the applied session (see below), it's important to get feedback on your completed UML class diagram before you leave the applied session. This way, you won't end up coding a misinterpreted or poorly designed solution at home.

What happens after today's applied session?

You have one week to continue working on your bootcamp tasks. Make sure to complete them **by the end of the day prior to your next applied session**.

We advise starting your UML design by sketching diagrams by hand (either on paper or using a digital sketching tool). This allows you to focus on understanding the details of UML syntax (e.g., arrowhead and line types). For your checkpoint, and especially for the final presentation, transfer your design to digital diagramming tools such as PlantUML, diagrams.net, or UMLet. While directly designing within a diagramming tool offers seamless updates, sketching by hand encourages deeper engagement with the design early in the development process. This approach helps ensure the design informs your implementation/coding, rather than the other way around, while the digital tools ultimately provide a clear and professional presentation.

Please, don't forget to check out the readings on EdLessons before each week's workshops and applied session. This week's readings include how to draw a UML class diagram, the difference between association and dependency relationships, and others that will help you when working on the bootcamp tasks.

Make sure to save your design documents, such as the UML class diagram, in the "docs" or "design" directory as a part of your project and add a prefix of "w2-" to the file name (e.g. "w2-uml.png"). Additionally, to ensure

your UML class diagram can be viewed from your GitLab repository, please save it as a PNG or PDF file.

UML Class diagrams are for humans!

Diagrams primarily serve human comprehension rather than strict adherence to the UML standard. While the UML standard offers a common graphical language, it's often unnecessary and not inherently advantageous to create models that strictly conform to it. These diagrams are intended for human interpretation and do not necessarily need to be machine-readable. Thus, the key consideration lies in ensuring clarity for the intended audience, rather than rigidly following UML conventions. Adhering to a shared graphical vocabulary, such as the UML notation, holds value insofar as it mitigates ambiguity and fosters effective communication between stakeholders.

Bootcamp REQ3: Implementation

Requirement 3: Implementation

Requirement 3a. Setting up the project and creating the Campsite class

To get started, clone the git repository and open it with your preferred IDE (we recommend IntelliJ IDEA). Create an `src` directory and mark it as “Sources Root” by right-clicking on the directory and selecting "Mark directory as > Sources Root". This lets IntelliJ know where your source codes are stored, enabling it to perform tasks such as code highlighting and error checking. If you are still not sure about how to do this, please check the instructions provided in the Applied Session on Week 1.

After that, add the following **Campsite** class that has a method named simulate.

```
public void simulate(){  1 usage
    System.out.println("Welcome to FIT2099 Camping Site");

    // The following method will be implemented in the next few requirements
    // camper.checkAllItems();

    System.out.println("Glad that you enjoy your time here!");
}
```

As of now, this `simulate` method of the Campsite class can only print a simple welcome message.

Next, create a new Application class with only a main method. Inside this method, create a **Campsite** object and call its simulate method. Finally, run your program to make sure everything is working as expected. You should be able to see the 2 string messages we put in your console.

At this moment, since the checkAllItems() method is not implemented yet, remember to comment out that line of code before testing your program, otherwise, errors will be thrown. Note the commented camper.checkAllItems() line in the screenshot above.

Suggestion: It's always a good idea to test your program in small steps, even if you have a complete design. For example, at this point, you should make sure the system runs and commit the changes to your Git repository with the changes following the recommended guidelines covered last week (see Supplementary Materials on Week 1). You can, for example, add the following commit comment:

Add initial Campsite and Application classes
- Implemented the Campsite class to manage camper-related functionality.
- Created the Application class as the main entry point for the system.

- Established the basic structure and initial method calls for loading and displaying item details.

Requirement 3b. The Item class

For this bootcamp, inside Campsite, there will be many items scattered around.

For the purpose of the bootcamp, we don't have a map, so just imagine they are scattered around.

Each item has its own name and weight.

What should the datatype for weight be? Should it be an integer? What about the name? Also, where can we initialise these values?

The Item class should also include a `toString()` method that combines the item's name and weight into a single string.

How can we define our own custom toString method? What is the purpose, and how does it work?

Just for testing purposes only: to ensure the Item class works correctly, you can modify Campsite's simulate() method to create an Item object and display its <<attribute>> using the `toString()` method. However, keep reading to understand why you might not want to create instances of the `Item` class from now on.

Hint: This requirement presents an excellent opportunity to delve deeper into formatting text output using the `String.format()` method. You can explore simple usage examples here, refer to the official documentation for detailed information here, or use a debugger to understand how it works.

Suggestion: After adding the Item class, it's a good time to **commit** the changes to git. Include a descriptive commit message indicating that you added a new class.

Requirement 3c. Item class's child classes

Let's discuss the items that we have prepared!

Each item in the **Campsite** has unique functionalities that we can use on our trip. To start, we've identified three types of Items: Bottle, Flint-and-Steel, and Bedroll. Note that these are item types, not item names. After skimming through these types, we find the following information.

- For Bottle (weight of 1kg), there's an additional attribute called capacity (1 Liter), which indicates how many liters of water the camper can drink from it.
- For Flint-and-Steel (weight of 0.5kg) and Bedroll (weight of 7.0kg), it doesn't have any additional attributes

Update on Aug 5th 15:17 : FlintAndSteel's weight is 0.5kg and Bedroll's weight is 7.0kg

Additional information regarding the name attribute, we can put like the brand, up to you.

Therefore, create three subclasses of the class Item, one for each respective type. Make sure you add their correct additional attribute according to the item type.

SPOILER ALERT: we will utilise the additional attributes mentioned above in the future, but we haven't gone to that stage yet so bear with us for a moment. There's no need to create other classes or anything not explicitly shown in this bootcamp.

Additionally, make sure to update the `toString` method for each subclass to include the particular attributes each has. For example, the `toString` method of the Bottle class should display the name, weight, and the capacity of the Bottle.

Since the Item class has already returned the name and weight. How can we avoid repetition here? You can learn more about **inheritance** and the '**extends**' keyword in this post-class material

Requirement 3d. Camper and Backpack

Now, let's add a new feature to our Campsite so that it can accommodate one camper. Technically, each camper will come prepared with **one** backpack with their items inside the backpack. Due to poor preparation, we only prepared three items inside our backpack.

To do this, let's add an array of Items to the Backpack class. Please, for this week, use **Java array, NOT ArrayList**, as we're not dealing with an arbitrary number of items. Make sure you understand the difference between Java Array and ArrayList, it will likely be a question asked by your TA during your interview.

Next, we will add (or provide the implementation for) the methods of the `Camper` that were mentioned below: `checkAllItems()`.

- The `checkAllItems()` method will show all the items that we have in the backpack, by showing all the details that we have in each item, such as its name, weight, and any additional information regarding the items (i.e. Capacity for Bottle, etc)

It is recommended to use a loop instead of repeated code and to avoid hardcoding the array length when we try to iterate through the list of items that we have inside the backpack.

Finally, run the `simulate()` method again to see whether you can get the same outputs as the expected output (provided later in this module).

We are not too strict about formatting so we don't require 100% identical output, but please keep in mind that it is certainly good practice to keep a clean and tidy format to demonstrate the information more clearly.

That's all for this week's bootcamp. Before you ask a TA to mark your work next week, make sure you have addressed the requirements and are happy with your work. Also, please do not leave the work to the last minute.

Checklist for Week 2 bootcamp:

Application:
- will run the simulate function that we get from Campsite

Campsite:
- accommodates one camper
- simulates the camping scenario by showing the welcome, all the items that camper has, and closing message.

Camper:
- has name and backpack
- Shows all the item that they have in the backpack

Backpack:
- Contains 3 items
- The items are Bedroll, Flint-and-Steel, and Bottle

Conditions for your bootcamp submission

> ⚠ **Be mindful of your bootcamp submission time as that depends on your allocated applied session time.**
> **Moodle submissions are not required as all bootcamp assessments will be conducted during the applied session.**

1. Last-Minute commits will not be considered

- Any commits made **during class** when the bootcamp is due for marking **will not** be considered.
- The version of your work at the official submission deadline is what will be assessed.
- Make sure all your work is committed and pushed **well before** the marking session begins.

2. Researching Previous Semester Works on GitHub

- Although we do not recommend this, you **are allowed** to refer to past semester's work by conducting your own research on GitHub. Please, note that the quality of any referenced work is **not guaranteed**—it could be a high-distinction submission or a failed attempt.
- This is an opportunity for you to learn how to **properly cite code**.
- You **must write all the code yourself** instead of copying and pasting.
- If you choose to build on someone else's work, you **must** ensure your contribution is original.
- If you are found copying code directly or excessively relying on another project, an **academic integrity alert** will be triggered.
- During your interview, you must show **a complete and solid understanding** of both the code you wrote and any referenced code. **Failure to do so will result in a score of 0 for the interview.**

3. Using ChatGPT & Proper Citation

- If you use ChatGPT to generate code, you **must** provide proper citations.
- Citations must be included in:
  - **In-line comments** within your code
  - **As per the citation guidelines for each assessment task in Moodle. For example:**

AI & Generative AI tools may be used in GUIDED ways within this assessment / task as per the guidelines provided.

In this task, AI can be used as specified for one or more parts of the assessment task as per the instructions.
Within this class, you are welcome to use foundation models (ChatGPT, GPT, DALL-E, Stable Diffusion, Midjourney, GitHub Copilot, and anything after) in a totally unrestricted fashion, for any purpose, at no penalty. However, you should note that all large language models still have a tendency to make up incorrect facts and fake citations, code generation models have a tendency to produce inaccurate outputs, and image generation models can occasionally come up with highly offensive products. You will be responsible for any inaccurate, biased, offensive, or otherwise unethical content you submit regardless of whether it originally comes from you or a foundation model. If you use a foundation model, its contribution must be acknowledged in the submission (in the form of a txt, pdf or word file titled FoundationModelsUsed added to your "docs" folder in GIT); you will be penalised for using a foundation model without acknowledgement.

Having said all these disclaimers, the university's policy on plagiarism still applies to any uncited or improperly cited use of work by other human beings, or submission of work by other human beings as your own. Moreover, missing contribution logs/GIT commits and/or failing any handover interview will be treated as a potential breach of academic integrity which will be further investigated.

Where used, AI must be used responsibly, clearly documented and appropriately acknowledged (see Learn HQ).

Any work submitted for a mark must:

1. represent a sincere demonstration of your human efforts, skills and subject knowledge that you will be accountable for.
2. adhere to the guidelines for AI use set for the assessment task.
3. reflect the University's commitment to academic integrity and ethical behaviour.

Inappropriate AI use and/or AI use without acknowledgement will be considered a breach of academic integrity.

- During your interview, you must demonstrate **a solid understanding** of any ChatGPT-generated code. **Failure to do so will result in a score of 0 for the interview.**

Marking rubric (Checkpoint)

In the **next** applied class, you will need to demonstrate progress through **a brief checkpoint review.** In this, you will showcase your development, progress and understanding of this week's bootcamp. Your teacher will use the following rubric to mark the progress of this bootcamp.

Req (1). GIT Commits (NOT graded this week)

Req (2). UML Design documents - clarity

Excellent - A design document is provided, is clear and syntactically correct. The design seems to fully address the requirements.

2 points

Satisfactory - A design document is provided and it seems to address most of the requirements. All required classes are present and the UML syntax is generally correct. Some key relationship(s) between classes are/is missing and there may be one or more minor syntax errors which do not affect the overall clarity of the documentation.

1 points

Unsatisfactory - A design document is not provided, OR the design does not address the requirements OR the documentation is unclear and not in accordance with UML syntax conventions or the application does not run OR Req (1) got 0 points.

0 points

Req (3). Implementation

Excellent - The implementation (the code) seems to fully address the requirements as per the UML design document (applying Object-Oriented principles).

2 points

Satisfactory - The implementation addresses the requirements with some design issues (not following the recommended design principles or the UML design document).

1 points

Unsatisfactory - The implementation does not really address the requirements even though some attempt was done or the application does not run OR Req (1) got 0 points.

0 points

Expected Output

Just to reiterate, the following messages are just for output indication, there's no need to produce identical outputs or use the numbers we use for weight.

```
Welcome to FIT2099 Camping Site

Here are the items that Cloudy has in the backpack:
Bedroll has weight of 7.00 kg - to rest.
Bottle has weight of 1.00 kg - to drink, with 1.0 liter left.
Flint and Steel has weight of 0.20 kg - to start a fire.

Glad that you enjoy your time here!

Process finished with exit code 0
```

In case that we want to put the name of the item attribute as the brand, then we can get like the following output.

```
Welcome to FIT2099 Camping Site

Here are the items that Cloudy has in the backpack:
Bedroll (KAMUI) has weight of 7.00 kg - to rest.
Bottle (Mountain Franklin) has weight of 1.00 kg - to drink, with 1.0 liter left.
FlintAndSteel (Aurora) has weight of 0.50 kg - to start a fire.

Glad that you enjoy your time here!

Process finished with exit code 0
```