Link to Video ;

https://drive.google.com/file/d/16J-wPLVyYnf7uBT_C5MgeoqSGODdeCEb/view?usp=share_link

Compile / run code

To run the code open terminal to unzipped folder. Type the following commands to get the gui to show:

- javac *.java
- java Bookstore

After typing these commands, the Gui will show.

- There are various fields that the user can type to alter the program:
- They can set the book shelve limit for each individual book shelve
- They can set the number of assistants.
- They can alter the mapping difference between ticks and milliseconds
- They can determine the number of book sections

There are two buttons at the bottom of the Gui for the user.

The first button is set print statements. There are 6 options;

1. Customer print statements – print out when and what genre the customer will buy
2. Assistant general print statements – General statements for assistants – collected, shelving statements
3. Delivery print statements – when books are deposited
4. Prioritize stocking statements – prints when books are prioritized
5. Bookshelf capacity statements – prints out when bookshelf capacity has been reached
6. Assistant break statements – prints out when assistants are taking a break

The second button is set Misc system. The user can;

- Set when inflation will go up
- How much profit bookstore will take
- How much money bookstore starts off on
- Set when the bookstore will close
- How long to close the bookstore

Once the values have been set the user can start the bookstore by pressing the button 'start bookstore'. They can see the print statements on the terminal and once the program has been finished the statements will be put out to a log file.

The number of days option in the Gui section will define the number of days the bookstore will run for. Once the bookstore has run for that number of days the program will stop and a data file will be output to the current directory showing the logs.

Thread Creation

The pattern used to create threads was to make all the threads be run inside the original bookstore class, this would help achieve consistency.

For the assistant threads, delivery threads and customer threads a class was created for each that would extend the thread class in java. A separate class for each thread would be called inside these threads. To start the threads the start function would be called within the bookstore class. For the assistant class an array of assistant threads is created dependent on the number of assistants defined in the Gui. Each assistant thread is started again within the bookstore class. For the delivery thread there is a while loop to create new deliveries. For the customer class there is an original class that loops to create new customer threads every 10 ticks on average.

patterns/strategies used to manage concurrency

There were a lot of issues to achieve concurrency when I began to write the program. This was especially the case for customers when they were trying to buy a book from a genre that wasn't there. It was hard originally to ensure other customers could still enter the bookstore and behave as normally while still ensuring that the customer who was waiting could buy the book when it became available. To achieve concurrency between these customer threads when this event happened synchronization would be implemented [1]. A monitor object for each genre was created so when a new genre came into the bookstore all customers who were waiting on that genre would be notified and allowed to buy the book that was available. As the customer threads are being constantly created inside a different class concurrency was achieved and other customers could enter the bookstore still while previous customers are waiting for a specific genre.

This effect of synchronization was used throughout the code. It was used also in the assistant thread, when the box of books is empty the assistant would wait through the method of synchronization, then when a box of books came in the assistant would be notified. An array of monitor objects with the size set to the number of assistants was created, to help achieve concurrency and ensure that the assistant starts right when a box of books comes in. Synchronization was also used to ensure that no more than one assistant takes books from the box.

HashMap had to be used inside the assistant thread for various functionality purposes such as defining the books that the assistant would take and the books that the assistant would put back. Originally concurrency was hard to achieve as the hash maps were not thread independent. To help achieve concurrency a thread local hash map object was created to ensure that for each thread the hash map would not change, and the code could execute properly [2].

Customer Functionality

Every 10 seconds a customer will enter the bookstore. They will randomly enter the bookstore every 10 seconds. To ensure this the thread will sleep randomly from 1-10 ticks then the thread will sleep for the remaining ticks that is calculated through the modulus function. There is a loop to create new customers. The customer will choose a random genre between all the genres they will wait if there are no books available and then take away a book from the book shelve and pay if the money system is on.

Assistant functionality

There will be the same number of assistant threads created as the number of assistants defined in the Gui. A loop keeps the assistant working. They will wait on other assistants and books coming in first. Then assistants will take breaks every x tick based on the Gui. The program will keep track of which assistants have taken a break to ensure that there is fairness for assistant breaks across all assistants. Based on the number of assistants books will be split evenly between assistants. The books available in the book box will be looped through, until there are no more books available, and they have been evenly dispersed throughout all assistants. Then the assistant that is stocking genres with the highest customer waiting list will take books from the box first. Once they have started to walk to the shelf the next assistant will take books from the box and so on and so forth.

The thread will sleep 10 ticks and also how many books the assistant is carrying to walk to a new section. If the bookshelves have reached the capacity that was defined in the Gui then there will be another thread local HashMap that will keep track of how many books the assistant will put back into the book box. It will take 10 ticks plus how many books the assistant brings back to walk to the book box. The assistant will then put the books back that they are carrying into the book box, then the loop will start again.

Deliver Books functionality

Similar to customer functionality, every 100 ticks on average a box of books will be delivered. A box of books will be randomly delivered every 100 ticks. The thread will sleep based on a random number between 1-100 the books will then be delivered and afterwards the thread will sleep until it reaches the next hundred tick mark based on the modulus function.

There will be 10 random books delivered based on a random genre. It will then notify any customers waiting for the genre delivered. Also, any assistants waiting on an empty box will also be notified.

Money System functionality

There will be a random price assigned to each book genre. Then every x number of ticks defined in the Gui inflation will increase by a random amount from 1-4 %. Based on the inflation percentage the prices will increase. Also, the profit percentage will be increased based on inflation to ensure that the bookstore is staying in line with the economy at the time. When a customer goes to buy a book the bookstore's money will increase based on the price of the genre of the box. When a box of books comes in the bookstore will buy the books if they have enough money to. When a customer then goes to buy a book, the bookstore will increase the price for the customer based on the profit percentage and the price of the books.

Close Store system functionality

There is a close store system that the user can either switch on or off inside the Gui. The close store tick count will inform the program when to close the store after how many ticks are defined inside that field. The total ticks store closed will define how long the store will stay closed for until it reopens up again.

Creation of the Gui

To create the Gui the jframe class was used.

Problems

One of the main problems that I ran into was the inaccuracy of the thread.sleep function [3]. It is not always accurate and can sleep either faster or slower than what is defined (usually faster) e.g., if you want to thread.sleep for 200 milliseconds it can sometimes only sleep for 195 milliseconds etc. For this reason, I would recommend running the program at a higher milliseconds per tick to get past this inaccuracy and get the most precise results. To counteract this, I used a while loop until it met the required number of ticks to pass and then slept the thread 1 tick until the required ticks was met.

Testing

To help test bookshelf capacity I would recommend using the assistant general print statements, delivery print statements and bookshelf capacity print statements.

References:

1. *Synchronization in Java* (2022) *GeeksforGeeks*. GeeksforGeeks. Available at: https://www.geeksforgeeks.org/synchronization-in-java/ (Accessed: March 12, 2023).
2. baeldung, W.by: (2022) *An introduction to threadlocal in Java*, *Baeldung*. Available at: https://www.baeldung.com/java-threadlocal (Accessed: March 12, 2023).
3. Pankaj (2022) *Thread.sleep() in Java - Java Thread sleep*, *DigitalOcean*. DigitalOcean. Available at: https://www.digitalocean.com/community/tutorials/thread-sleep-java (Accessed: March 12, 2023).