

Manuel Utilisateur

Simulation du project Producteur-Consommateur

Table of material

I. Apercu

II. Avant Compiler

III. Compile & Run du project

VI. Probleme Communs

V. Contact

I. Aperçu

Ce projet démontre le problème classique du Producteur-Consommateur en utilisant Java. Le problème du Producteur-Consommateur est un problème de synchronisation qui consiste à s'assurer que les producteurs (qui produisent des données et les ajoutent à un tampon) et les consommateurs (qui prennent des données du tampon) fonctionnent sans rencontrer de conditions de course. Ce projet utilise les méthodes synchronisées de Java. Il comprend une interface graphique (GUI) pour simuler la production et la consommation de produits.

Caractéristiques :

1. Les producteurs peuvent ajouter des données au tampon tant qu'il y a un espace vide.
2. Les consommateurs peuvent consommer des données du tampon si elles existent.
3. La GUI permet aux utilisateurs de :
4. Ajouter un nouveau producteur.
5. Ajouter un nouveau consommateur.
6. Supprimer un producteur.
7. Supprimer un consommateur.
8. Mettre à jour la taille du tampon.
9. Voir le nombre de producteurs et de consommateurs existants.

II. Avant Compiler

1. telecharger git

```
$ sudo dnf install git-all
```

2. Clone the repository using the following command:

```
git clone https://github.com/yourusername/producer-consumer-java.git  
cd producer-consumer-java
```

Pour cloner le dépôt, utilisez la commande suivante : ``git clone https://github.com/yourusername/producer-consumer-java.git``. Cette commande téléchargera une copie locale du projet sur votre machine. Ensuite, naviguez dans le répertoire du projet avec ``cd producer-consumer-java`` pour accéder aux fichiers et commencer à travailler sur le projet.

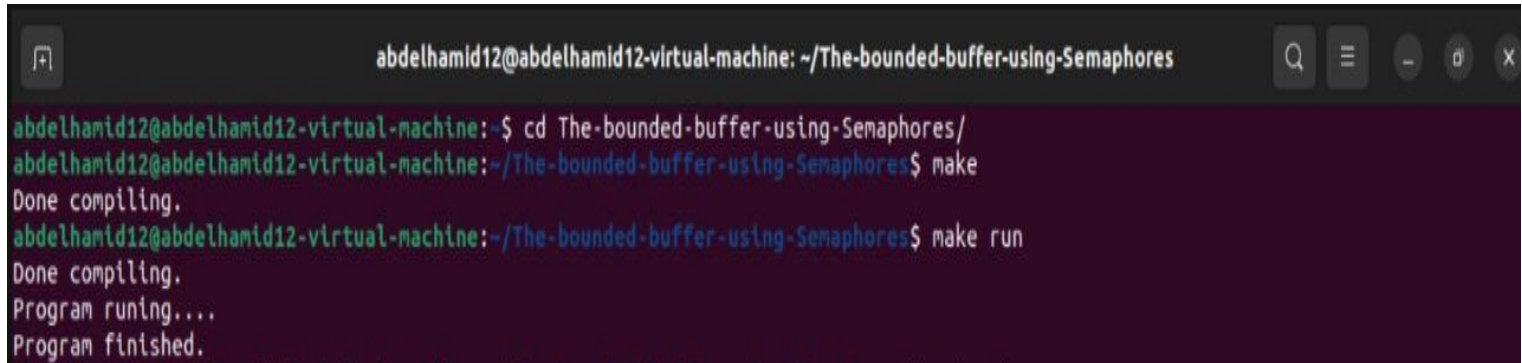
3. install JDK

```
sudo apt update  
sudo apt install default-jdk
```

``sudo apt update`` met à jour la liste des packages disponibles. ``sudo apt install default-jdk`` installe le kit de développement Java (JDK) par défaut. Le JDK est nécessaire pour compiler et exécuter des programmes Java.

III. Compile & Run du project

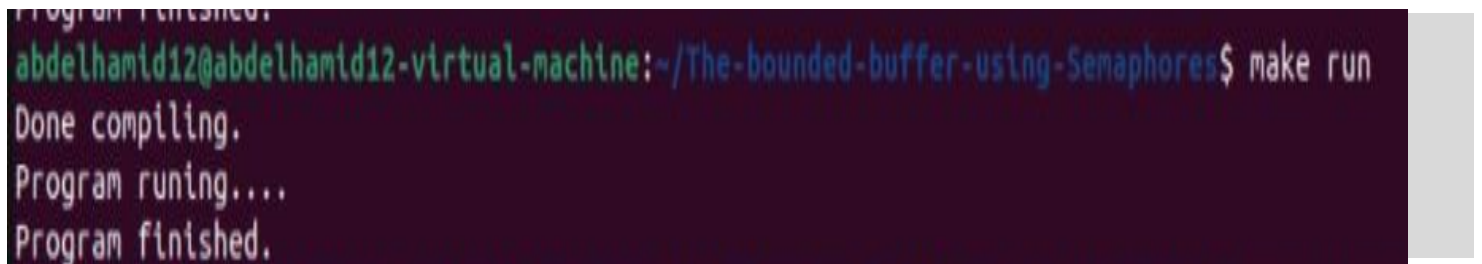
1. Naviguez jusqu'au répertoire du projet :

A terminal window with a dark background. The title bar shows the user 'abdelhamid12' on a 'virtual-machine' at the directory '~/The-bounded-buffer-using-Semaphores'. The terminal shows the following commands and output:

```
abdelhamid12@abdelhamid12-virtual-machine: ~/The-bounded-buffer-using-Semaphores
abdelhamid12@abdelhamid12-virtual-machine: $ cd The-bounded-buffer-using-Semaphores/
abdelhamid12@abdelhamid12-virtual-machine:~/The-bounded-buffer-using-Semaphores$ make
Done compiling.
abdelhamid12@abdelhamid12-virtual-machine:~/The-bounded-buffer-using-Semaphores$ make run
Done compiling.
Program runing....
Program finished.
```

répertoire du projet. La deuxième commande, `make`, est utilisée pour exécuter les instructions de compilation ou d'autres tâches définies dans le fichier `Makefile` du projet. L'ensemble de ces deux commandes permet donc de se déplacer vers le répertoire du projet et d'exécuter les tâches de construction du projet à l'aide de `make`.

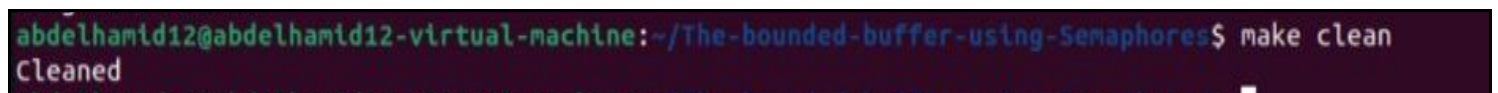
2. Running the Project

A terminal window showing the execution of the 'make run' command. The output is as follows:

```
abdelhamid12@abdelhamid12-virtual-machine:~/The-bounded-buffer-using-Semaphores$ make run
Done compiling.
Program runing....
Program finished.
```

configurés avec un fichier `Makefile` pour exécuter le programme principal ou l'application du projet. Cette commande recherche généralement une cible appelée `run` dans le `Makefile`, qui contient les instructions pour exécuter l'application. Elle peut inclure la compilation des fichiers source si nécessaire, suivie de l'exécution du programme résultant.

3. Cleaning the project

A terminal window showing the execution of the 'make clean' command. The output is as follows:

```
abdelhamid12@abdelhamid12-virtual-machine:~/The-bounded-buffer-using-Semaphores$ make clean
Cleaned
```

La commande `make run` est une commande couramment utilisée dans les projets configurés avec un fichier `Makefile` pour exécuter le programme principal ou l'application du projet. Cette commande recherche généralement une cible appelée `run` dans le `Makefile`, qui contient les instructions pour exécuter l'application. Elle peut inclure la compilation des fichiers source si nécessaire, suivie de l'exécution du programme résultant.

1. Fichier Makefile

```
1  # Define variables
2  SRC_DIR = The-bounded-buffer-problem
3  SRC_FILES = $(wildcard *.java)
4  CLASS_FILES = $(SRC_FILES:.java=.class)
5  MAIN_CLASS = MainSemaphores
6  # MAIN_CLASS = MainBlockingQueue
7
8  # Default target
9  all: $(CLASS_FILES)
10     @echo "Done compiling."
11  # Rule to compile all .java files to .class files
12  $(CLASS_FILES): $(SRC_FILES)
13     javac $(SRC_FILES)
14
15  # Run the main class
16  run: all
17     @echo "Program runing....."
18     java $(MAIN_CLASS)
19     @echo "Program finished."
20
21  # Clean up .class files
22  clean:
23     rm -f *.class
24     @echo "Done Cleaning."
25
26  # Phony targets
27  .PHONY: all run clean
```

2. Interface

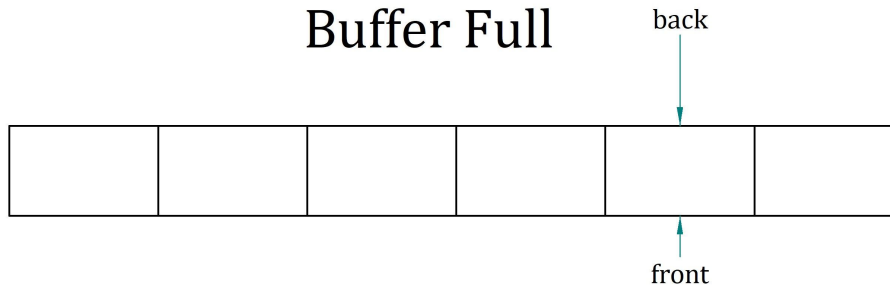


Producer Consumer Simulation

Added producer. Total producers: 1
Produced: 14 by Producer : 30
Produced: 56 by Producer : 30
Produced: 79 by Producer : 30
Produced: 8 by Producer : 30
Produced: 57 by Producer : 30
Produced: 22 by Producer : 30
Produced: 58 by Producer : 30
Produced: 94 by Producer : 30
Produced: 41 by Producer : 30
Produced: 33 by Producer : 30
Buffer is full
Removed producer. Total producers: 0
Added consumer. Total consumers: 1
Consumed: 14 by Consumer : 31
Produced: 12 by Producer : 30
Buffer is full
Consumed: 56 by Consumer : 31
Consumed: 79 by Consumer : 31
Consumed: 8 by Consumer : 31
Consumed: 57 by Consumer : 31
Consumed: 22 by Consumer : 31
Consumed: 58 by Consumer : 31
Consumed: 94 by Consumer : 31
Consumed: 41 by Consumer : 31
Consumed: 33 by Consumer : 31
Consumed: 12 by Consumer : 31
Buffer is empty

VI. Probleme Communs

1. Buffer Full:

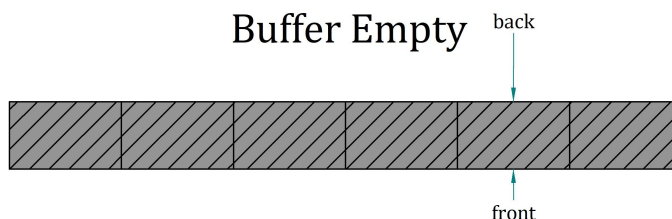


- Issue: Lorsque le tampon est plein, cela signifie qu'il n'y a plus d'espace disponible pour ajouter de nouvelles données produites.

- Explanation : Cela peut entraîner un blocage de la production car le producteur ne peut pas ajouter de nouvelles données au tampon, ce qui peut entraîner une perte potentielle de données si elles ne sont pas traitées immédiatement ou si aucune action n'est prise pour gérer le problème.

- Resolution: Pour résoudre ce problème, une stratégie de gestion des données peut être utilisée, telle que l'attente jusqu'à ce qu'un espace se libère dans le tampon ou la suppression des données les plus anciennes pour faire de la place pour de nouvelles données.

2. Buffer Empty:



- Issue: Lorsque le tampon est vide, cela signifie qu'il n'y a pas de données disponibles pour être consommées.

- Explanation: Cela peut entraîner un blocage de la consommation car le consommateur ne peut pas prendre de données du tampon, ce qui peut entraîner une interruption du traitement ou un dysfonctionnement du système si aucune mesure n'est prise pour gérer le problème.

- Resolution: Pour résoudre ce problème, une stratégie de gestion des données peut être utilisée, telle que l'attente jusqu'à ce que des données soient produites et ajoutées au tampon ou l'envoi d'une notification pour indiquer que le tampon est vide.

Contact

Pour toute assistance supplémentaire, veuillez contacter :

Emails :

abdelhamidelmadani45@gmail.com

boutainaotaku2003@gmail.com

imaneelghabipro@gmail.com

ilyasagaz0@gmail.com