Defining the Challenge - Producer Consumer Problem

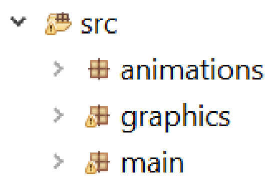
The Producer Consumer problem (otherwise referred to as the Bounded Buffer) is a classic Concurrency problem described by Dijkstra in 1965.[1] The premise being multiple processes producing data that is stored in a common structure referred to as the Buffer. Meanwhile, a consumer takes data from the Buffer for consumption. The challenge of the problem lies in synchronizing the processes with several overarching expectations:[2]

1. A producer should only produce if the buffer is not full
2. Data can only be consumed if the buffer is not empty
3. Access to the buffer should not be done at the same time for both a consumer and producer

Code Specifications

This project was coded in Java using the Eclipse IDE. The ModulePath in the Java Build Path relied on Java's JDK 14.0.2. Meanwhile, the ClassPath was supplied by Professor Prasun Dewan through a provided Jar file. It is necessary to utilize the ClassPath Jar file to display graphics through Professor Dewan's ObjectEditor Software Program.

Packages Breakdown



The project source file is split into three packages: animations, graphics, and main. Graphics contain a shared location for all classes and interfaces that build the properties of the program in addition to each property's functionality. Animations contain the composers and animators that allow for movement of the defined properties defined in graphics. Main sets the scene and calls on ObjectEditor to begin compiling the program.

Graphics

The graphics package contains the following list of classes and interfaces:

1 https://jcsites.juniata.edu/faculty/rhodes/os/ch5b.htm
2 https://www.javatpoint.com/producer-consumer-problem-in-os

Class Name	**Interface Name**	**Properties**	**Editable?**	**Comments**
ABell	Bell	bell, text	yes	
ABoundedBuffer	BoundedBuffer	circle, lineOne, lineTwo, lineThree, lineFour, lineFive, lineSix, lineSeven, lineEight	no	The lines are used to define the legs and surface of the buffer circle
AChef	Chef	chef, table	no	
AnAngle	Angle	left, right	no	Defines arms and legs of the avatars
AnAvatar	Avatar	arms, legs, torso, head, text	yes	
APatron	Patron	paton, table	no	
APlate	Plate	plate, text	yes	
AProducerConsumerScene	ProducerConsumerScene	patronList, chefList, buffer, plateList, bellsList, queueList	no	Sets the scene for the program
ATable	Table	horizontalTop, horizontalBottom, verticalLeft, verticalRight, legOne, legTwo, legThree, legFour	no	

Main Package

The main package contains the Main method to run the program. Inside the animateScene() method, we define the method attributes we want to appear as buttons on the top of our OE

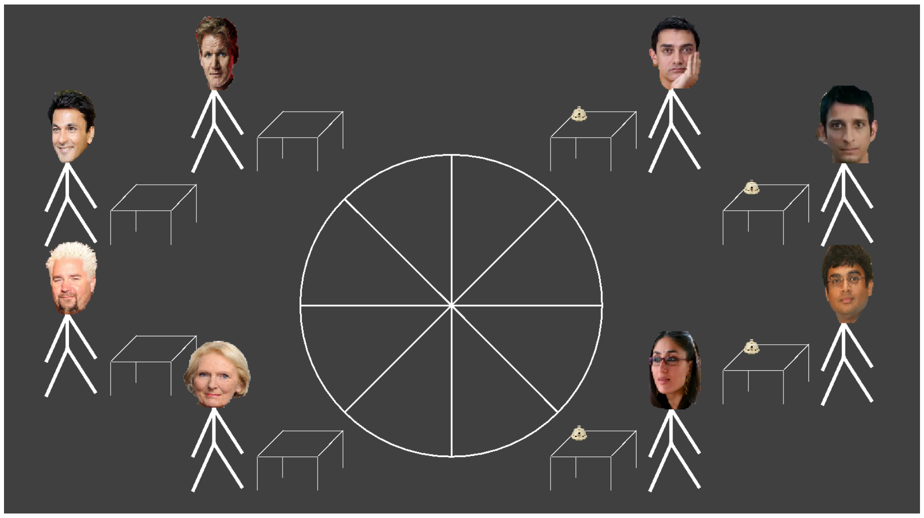
1 https://jcsites.juniata.edu/faculty/rhodes/os/ch5b.htm
2 https://www.javatpoint.com/producer-consumer-problem-in-os

window. The try-catch block defaults the number of patrons and chefs as four. Finally, the OE Window is defined in color and size.

Animations

The animations are defined in an animator-command breakdown. The command takes in values for the animation, including the pause time and breaking condition, and passes it to the animator. The animator contains the logic for how the graphics will continue to update and stop once reaching the breaking condition.

InitScene Default Setup



1 https://jcsites.juniata.edu/faculty/rhodes/os/ch5b.htm
2 https://www.javatpoint.com/producer-consumer-problem-in-os