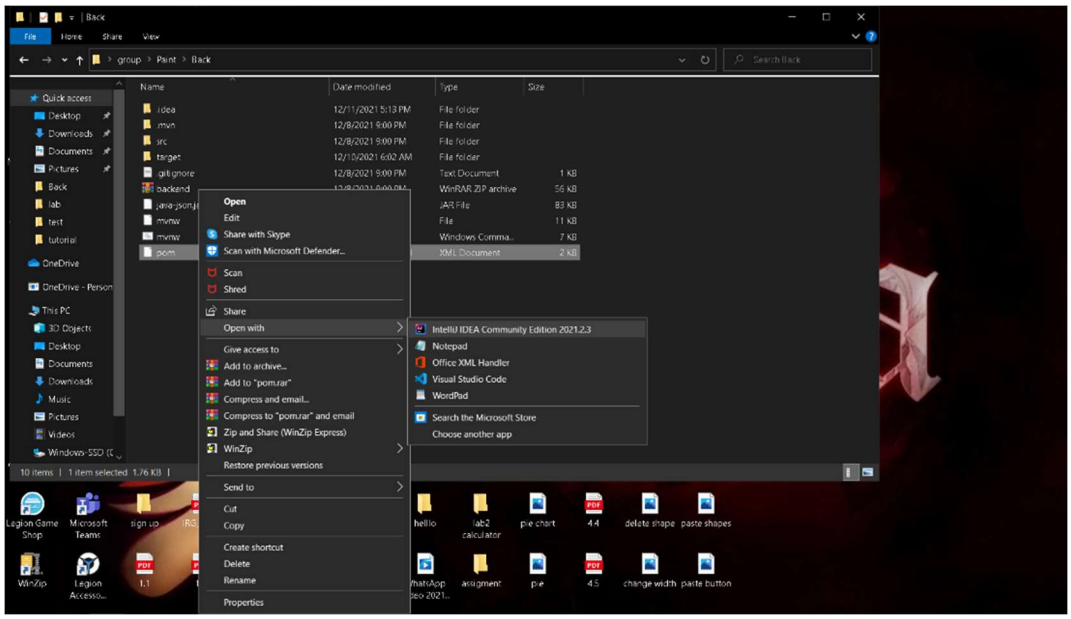
Assignment #5 Report
Product Consumer

No	Name	ID
1	عبدالرحمن السيد احمد علي	19015893
2	عبدالرحمن السيد جاد السيد	19015894
3	عبدالعزيز محمد عبدالعزيز محمد	19015941
4	عمر خيرت محمد ابو ضيف	19016063

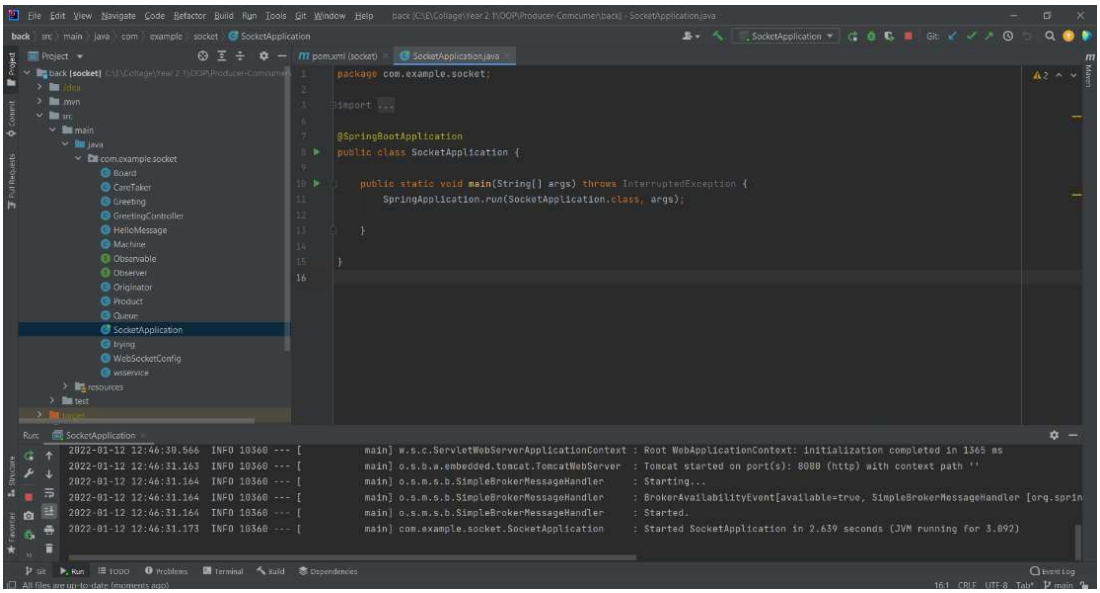
1. How to run the program

a. Run the backend application:

- Open the “Back” folder.
- Open the folder with IntelliJ or Eclipse IDEA.

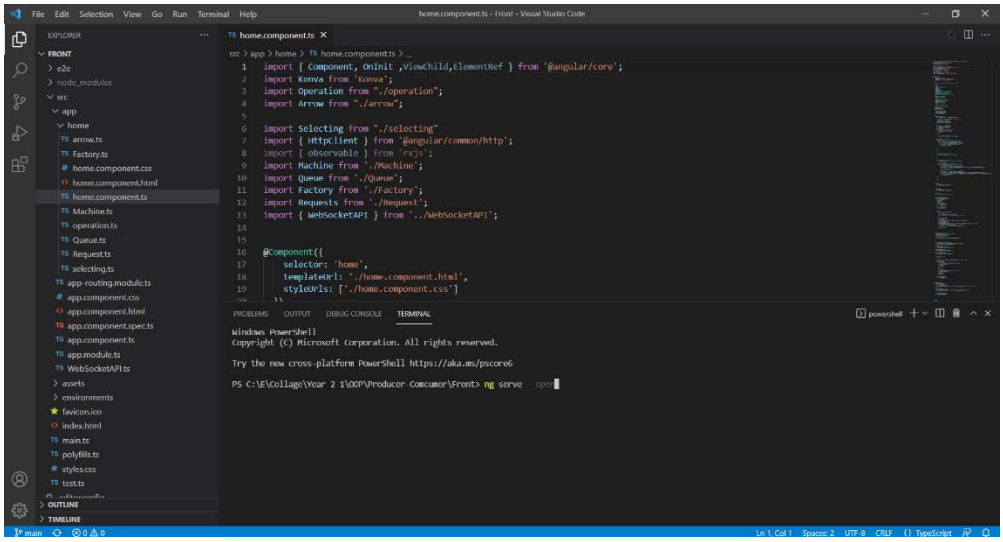


- Run the backend server from running SocketApplication.



b. Run the frontend application:

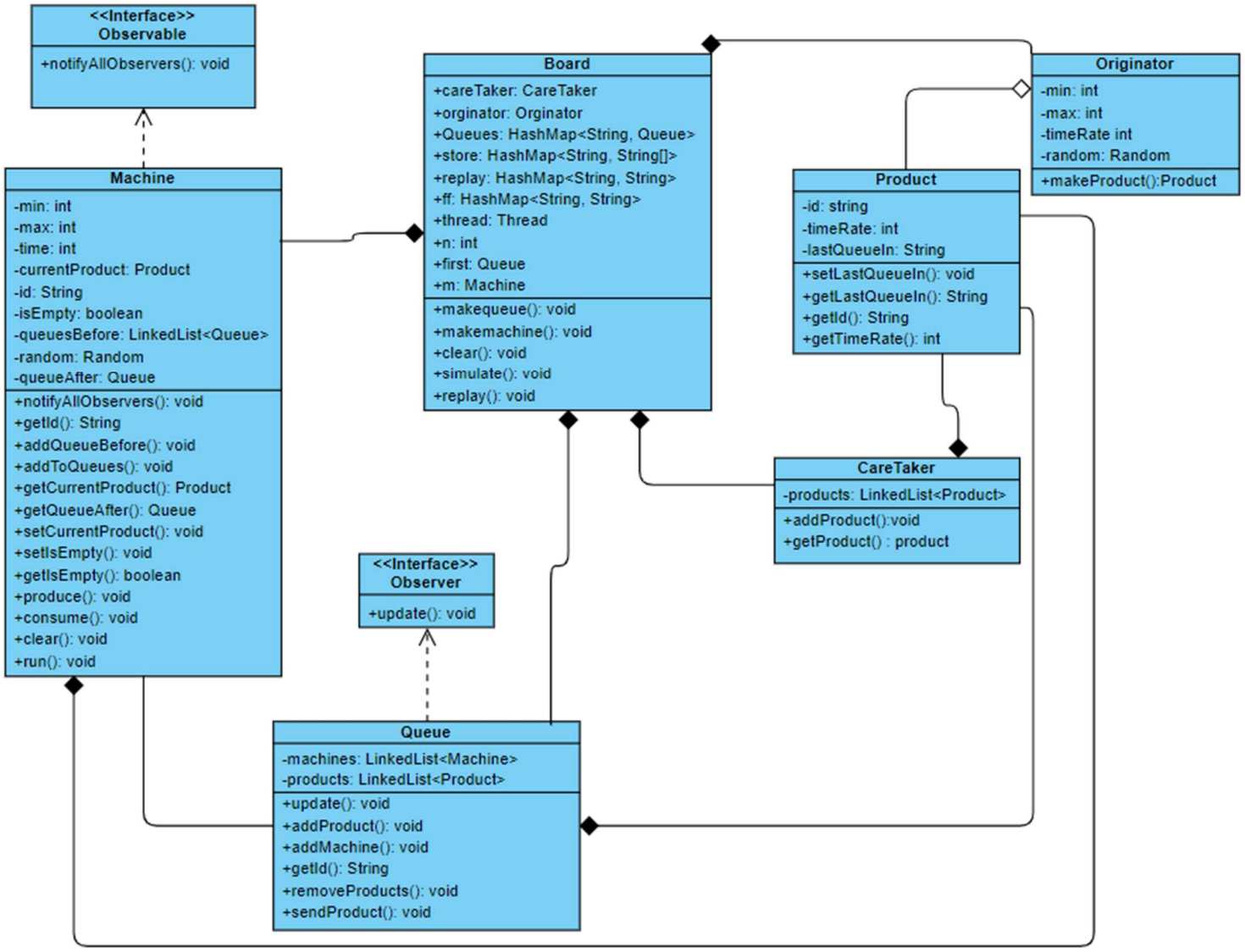
- Open the “Front” folder.
- Open the folder with Visual Studio Code IDEA.
- From the IDEA, open the terminal menu then chose “new terminal”
- Write the command “ng serve --open”.
- If it said:” Port 4200 is already in use. Would you like to use a different port?”, just type “y”.



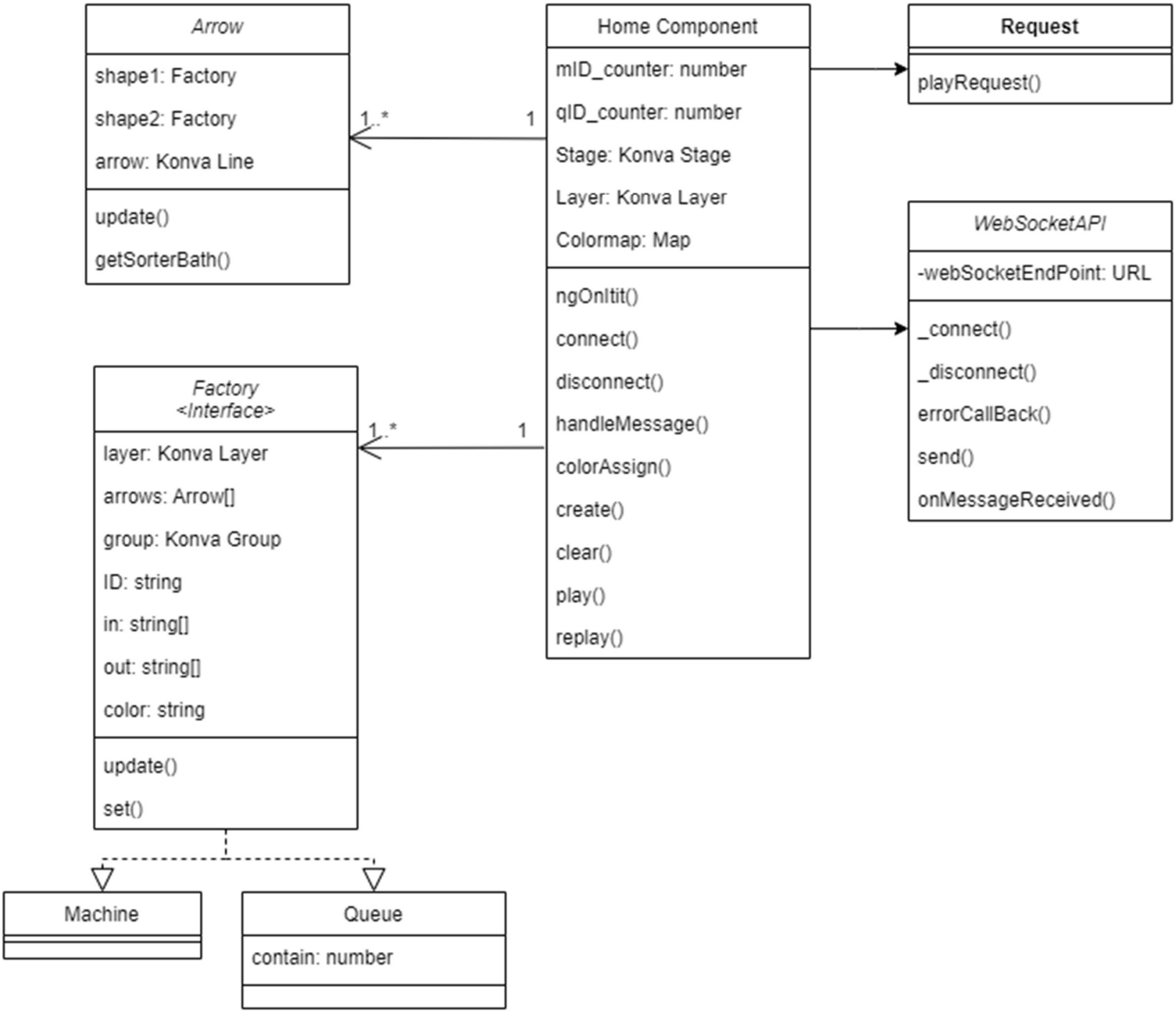
c. The application opens your default browser.

2. UML:

a. Back UML:



b. Front UML:


Arrow
shape1: Factory
shape2: Factory
arrow: Konva Line
update()
getSorterBath()
1..*
1
Home Component
mID_counter: number
qID_counter: number
Stage: Konva Stage
Layer: Konva Layer
Colormap: Map
ngOnItit()
connect()
disconnect()
handleMessage()
colorAssign()
create()
clear()
play()
replay()
Request
playRequest()
WebSocketAPI
-webSocketEndPoint: URL
_connect()
_disconnect()
errorCallBack()
send()
onMessageReceived()
Factory
<Interface>
layer: Konva Layer
arrows: Arrow[]
group: Konva Group
ID: string
in: string[]
out: string[]
color: string
update()
set()
1..*
1
Machine
Queue
contain: number

3. Design Patterns Applied:

A. In Back:

Producer -consumer: in which is it used to coordinate the asynnchrouns production and consumption of information.

```
public void produce() throws InterruptedException
{
    rec.put("product",this.currentProduct.getId());
    rec.put("out",this.currentProduct.getLastQueueIn());
    rec.put("in",this.id);
    this.tg.send2(rec);
    this.isEmpty=false;
    Timer timer = new Timer();
    timer.schedule(() -> {
            try
            {
                consume();
            }
            catch (InterruptedException e)
            {
                e.printStackTrace();
            }
    }, time);
```

```
public void consume() throws InterruptedException
{
    this.isEmpty = true;
    this.queueAfter.addProduct(this.currentProduct);
    String in ;
    String out;
    String product;
    out=getId();
    product= this.currentProduct.getId();
    in=this.currentProduct.getLastQueueIn();

   sent.put("product",product);
    sent.put("in",in);
    sent.put("out",out);
  this.tg.send2(sent);
    this.thread.join();
    this.notifyAllObservers();
```

2. observer:

Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically

```
public interface Observer
{
    public abstract void update( ) throws InterruptedException;
}
```

```
public interface Observable
{
    void notifyAllObservers() throws InterruptedException;
}
```

3.Snapshot:

It is used to restore state of an object to a previous state.

```
package com.example.socket;

import java.util.LinkedList;

public class CareTaker
{
    LinkedList<Product> products;
    public CareTaker() { products = new LinkedList<Product>(); }
    public void addProduct(Product product) { products.addLast(product); }
    public Product getProduct()
    {
        Product product = products.getFirst();
        products.removeFirst();
        products.addLast(product);
        return product;
    }
}
```

B. In Front:

- **Factory** Design Pattern: in creating the Machine or the Queue.

Factory.ts

src > app > home > Factory.ts > ...

```ts
import Konva from "Konva";
import Arrow from "./arrow";

export interface  Factory{
    layer: Konva.Layer
    arrows: Arrow[]
    machineGroup: Konva.Group
    ID:string
    inn: string[]
    out: string[]
    color: string
    update(x:string):void
    set(x:number):void
}
export default Factory;
```

Machine.ts

src > app > home > Machine.ts > Machine

```ts
import Konva from "Konva";
import Arrow from "./arrow";
import Factory from "./Factory";

class Machine implements Factory{
    layer: Konva.Layer
    arrows: Arrow[] = []
    machineGroup: Konva.Group
    ID!:string
    inn: string[] =[]
    out: string[]= []
    color = 'red'

    constructor(layer: Konva.Layer, shift:number, m:number){
        this.layer = layer
        var shp = new Konva.Group({
            x: 150+shift,
            y: 150+shift,
            width: 130,
            height: 25,
            rotation: 0,
            draggable: true,
            name:"Machine",
            id: "m"+m

        })

        shp.add(new Konva.Circle({

            radius:75/2,
            stroke: "rgb(0,0,0)",
```

Machine.ts

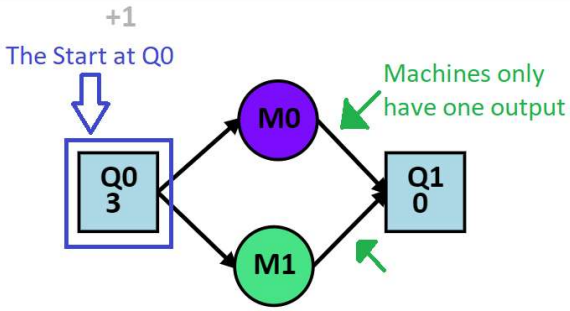
src > app > home > Machine.ts > Machine

```ts
import Konva from "Konva";
import Arrow from "./arrow";
import Factory from "./Factory";

class Machine implements Factory{
    layer: Konva.Layer
    arrows: Arrow[] = []
    machineGroup: Konva.Group
    ID!:string
    inn: string[] =[]
    out: string[]= []
    color = 'red'

    constructor(layer: Konva.Layer, shift:number, m:number){
        this.layer = layer
        var shp = new Konva.Group({
            x: 150+shift,
            y: 150+shift,
            width: 130,
            height: 25,
            rotation: 0,
            draggable: true,
            name:"Machine",
            id: "m"+m

        })

        shp.add(new Konva.Circle({

            radius:75/2,
            stroke: "rgb(0,0,0)",
```

4. Design decisions and Assumptions:

A. The first queue must be Q0 and that's the start of simulation.

B. The machines can only have one output.



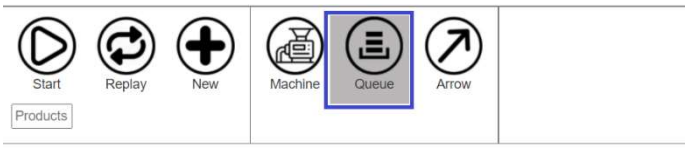
C. The design must start with Q0 and end with any queue,
Can not end with machine

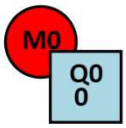
D. Every machine takes random time to finish the product from 3 to 8 second.

E. At the start the entered number of products enter the Q0 at random time rate between 2 and 5 seconds.

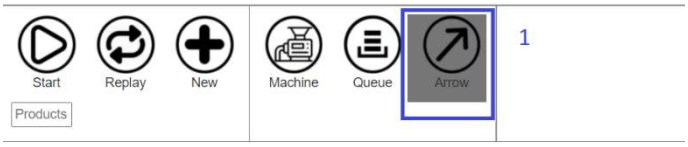
F. User Guide:

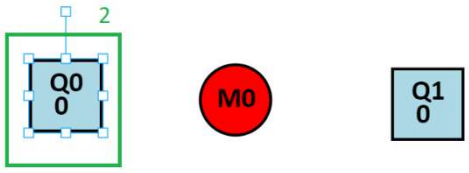
A. To make a **Machine** or a **Queue**, simply click on the machine or the queue button.

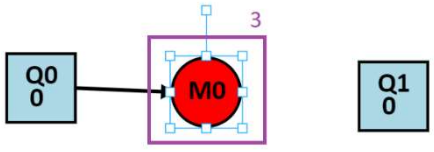




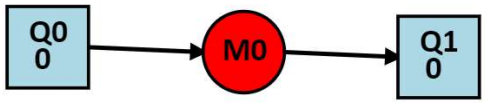
B. To make an **arrow**, first click on the arrow button then select the first element then the second element, and the arrow will be drawn from the first to the second.



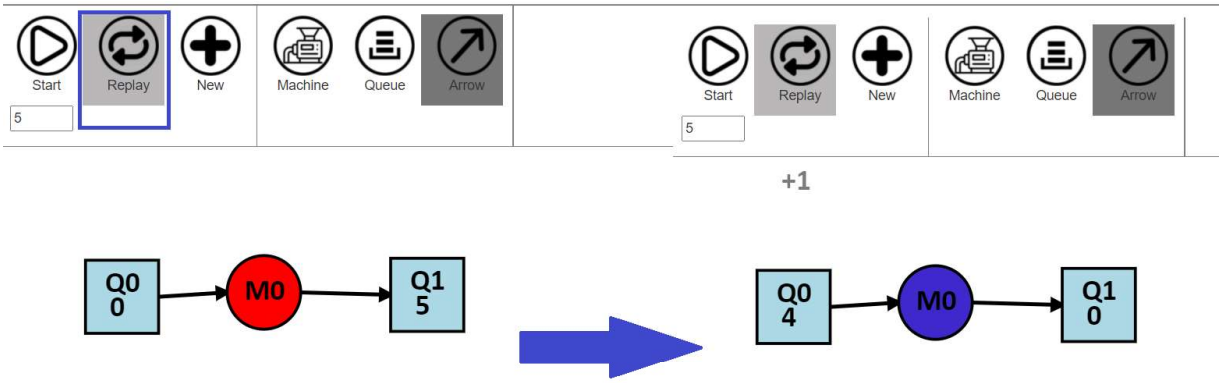




C. To **Start** the simulation, you must enter the number of products first then click start.



D. To **Replay**, after the simulation end click on replay button.



E. To start **new** design, just click on new