MAKERERE UNIVERSITY

COLLEGE OF COMPUTING AND INFORMATION SCIENCES

DEPARTMENT OF NETWORKS

BACHELOR OF SCIENCE IN SOFTWARE ENGINEERING

WEEK 4 REPORT FOR KIGGS SUPERMARKET

FOR

PREPARED BY BSE23-4

NAME	REGISTRATION NUMBER	STUDENT NUMBER
MUGERWA JOSEPH	19/U/0210	1900700210
MULIKATETE ANGELLA	19/U/26906/EVE	19007026906
NAMIIRO HABIIBAH	19/U/0216	1900700216
MUGAMBA BRUNO M.H	19/U/8859/EVE	1900708859

Strategy Pattern

The Strategy pattern aims at encapsulating functionalities and making them interchangeably. **PaymentMethod** is an interface used to define the functionality required by the different gateways and has pay() method that will be used to handle payment.

```
export interface PaymentMethod {
 pay(amount: number): void;
}
```

We have two classes that implement this interface **MTN** and Airtel. These two classes take in the phone and pin.

```
 class MTN implements PaymentMethod {
   constructor(private readonly phone: number, private readonly pin: number) {}
    pay(amount: number): void {
     console.log(`Paying ${amount} with MTN mobile money`);
   }
 }
```

The above shows the implementation of the pay method with MTN.

```
 class Airtel implements PaymentMethod {
   constructor(private readonly phone: number, private readonly pin: number) {}
   pay(amount: number): void {
     console.log(`Paying  ${amount} with Airtel mobile money.`);
   }
 }
```

The above shows the implementation of the pay method with Airtel.

Observer pattern

```
export interface Observer {
 notify: (productId: ProductInterface) => void;
 update: (productId: string) => void;
}
```

In this pattern we basically have an interface called Observer that defines two methods notify and update where the update() takes the string that resulted after scanning.

```
// Helper function to look up a product from a barcode
function getProductFromBarcode(barcode: string) : ProductInterface {
 let p = new ProductFactory()
 let cart = new CartService();

  let product = {
     productName: 'Dell XPS3',
     productPrice: 1200,
     dateOfPurchase: Date.now(),
     productImage: '',
     productDescription: '13-inch laptop with Intel Core i7 processor',
     productId: 1,
     productCategory: 'Electronics',
     productQuantity: 3,
   };
let products :ProductInterface[] = [
 p.createProduct(product),
];
  return products.filter((product) => product.productName === barcode)[0];
}
```

Unit tests for Strategy pattern

```
test("should pay with Airtel gateway", () => {
     const strategy = new Airtel(0758743490, 2580);
     const paymentGateway = new PaymentGateway(strategy);
     const logSpy = jest.spyOn(console, "log");

     paymentGateway.pay(100);

```

```
    expect(logSpy).toHaveBeenCalledWith("Paying 100 with Airtel");
  });
```
From the code snippet above we are testing whether payment with airtel is successful.

```
test("should set gateway", () => {
    const mtn = new MTN(0778743490, 2580);
    const paymentGateway = new PaymentMethod(creditCardStrategy);

    const airtelGateway = new Airtel(0750482089, 2580);
    paymentGateway.setGateway(airtelGateway);
    expect((paymentGateway).method).toBe(airtelGateway);
  });
```
The code snippet above tests if a payment gateway is actually set.