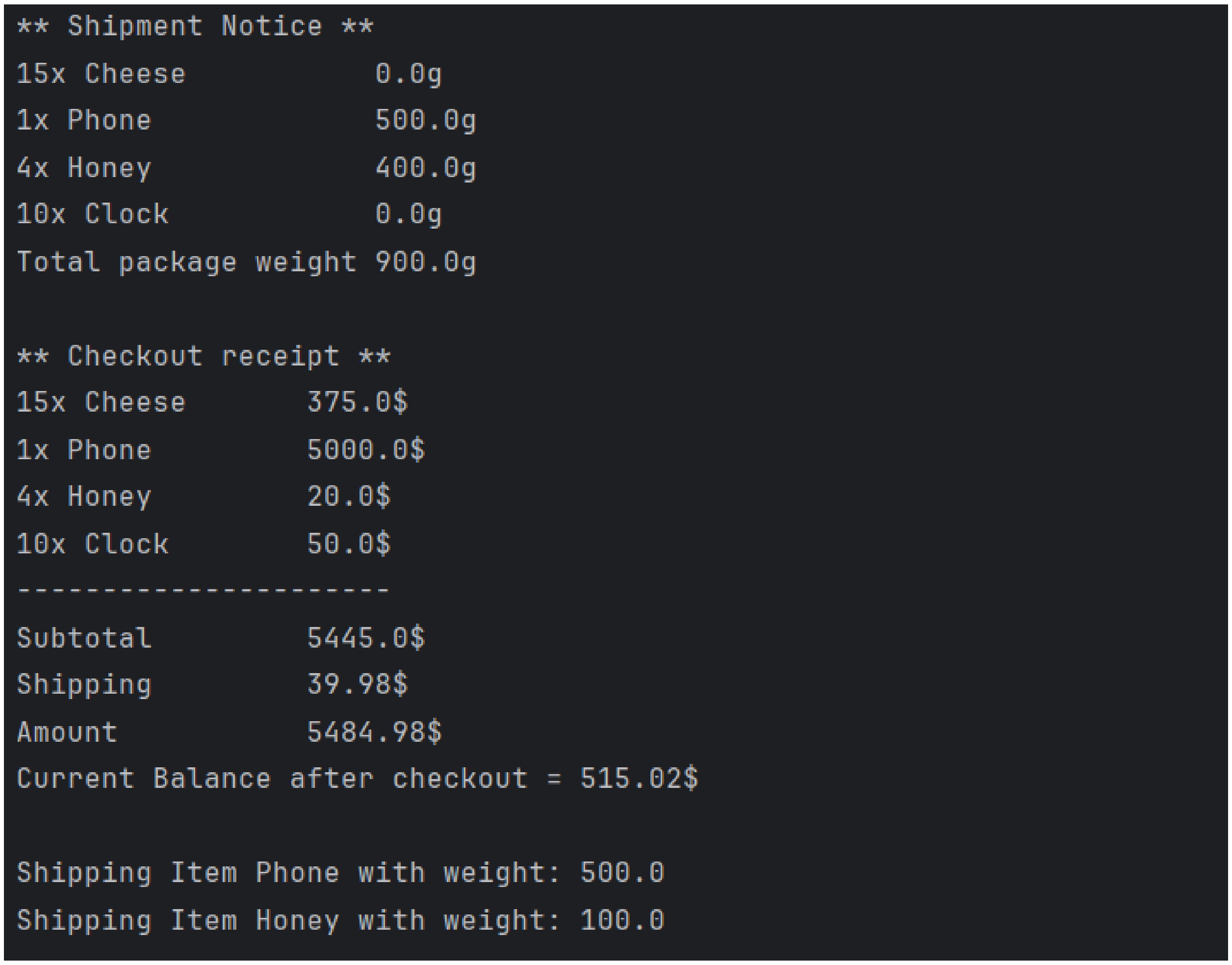
Testing Code Examples

Code (adding diffe):

```java
Product cheese = new Product(
        "Cheese",
        25.0F,
        15,
        new NotShippable(),
        new Expirable(2026)
);

Product phone = new Product(
        "Phone",
        5000.0F,
        1,
        new Shippable(500,19.99F),
        new NotExpirable()
);

Product honey = new Product(
        "Honey",
        5.0F,
        8,
        new Shippable(100,19.99F),
        new NotExpirable()
);

Product clock = new Product(
        "Clock",
        5.0F,
        100,
        new NotShippable(),
        new NotExpirable()
);

// Creating a Cart
Cart cart = new Cart(6000);

// Adding Items to cart
cart.add(cheese, 15);
cart.add(phone, 1);
cart.add(honey, 4);
cart.add(clock, 10);

cart.checkout();

System.out.println();
// Printing all shippable items using ShippingService
for (SimpleShippableI shippableItem : ShippingService.toBeShipped){
    System.out.println("Shipping Item " + shippableItem.getName() + " with
weight: " + shippableItem.getWeight());
}
```

Output:



```
** Shipment Notice **
15x Cheese           0.0g
1x Phone             500.0g
4x Honey             400.0g
10x Clock            0.0g
Total package weight 900.0g

** Checkout receipt **
15x Cheese       375.0$
1x Phone         5000.0$
4x Honey         20.0$
10x Clock        50.0$
----------------------
Subtotal         5445.0$
Shipping         39.98$
Amount           5484.98$
Current Balance after checkout = 515.02$

Shipping Item Phone with weight: 500.0
Shipping Item Honey with weight: 100.0
```

Code (adding 2 phones while having only 1 in stock):

```java
package org.example;

public class Main {
    public static void main(String[] args) {
        try {
            Product cheese = new Product(
                    "Cheese",
                    25.0F,
                    15,
                    new NotShippable(),
                    new Expirable(2026)
            );

            Product phone = new Product(
                    "Phone",
                    5000.0F,
                    1,
                    new Shippable(500,19.99F),
                    new NotExpirable()
            );

            Product honey = new Product(
                    "Honey",
                    5.0F,
                    8,
                    new Shippable(100,19.99F),
                    new NotExpirable()
            );

            Product clock = new Product(
                    "Clock",
                    5.0F,
                    100,
                    new NotShippable(),
                    new NotExpirable()
            );

            // Creating a Cart
            Cart cart = new Cart(6000);

            // Adding Items to cart
            cart.add(cheese, 15);
            cart.add(phone, 2);
            cart.add(honey, 4);
            cart.add(clock, 10);

            cart.checkout();

            System.out.println();
            // Printing all shippable items using ShippingService
            for (SimpleShippableI shippableItem :
ShippingService.toBeShipped){
                System.out.println("Shipping Item " + shippableItem.getName()
+ " with weight: " + shippableItem.getWeight());
            }


        } catch (Exception e) {
            System.out.println(e.getMessage());
        }

    }
}
```

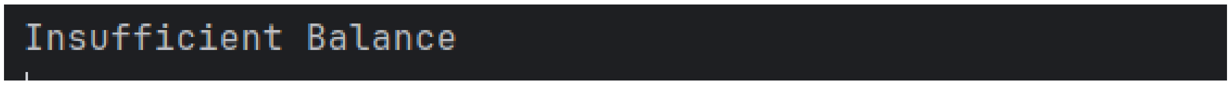
Output:

```
Phone:: Not enough stock available
```

Code (adding same products but with no sufficient balance):

```java
package org.example;
```

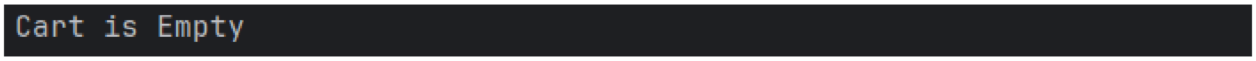
```
public class Main {
    public static void main(String[] args) {
        try {
            Product cheese = new Product(
                    "Cheese",
                    25.0F,
                    15,
                    new NotShippable(),
                    new Expirable(2026)
            );

            Product phone = new Product(
                    "Phone",
                    5000.0F,
                    1,
                    new Shippable(500,19.99F),
                    new NotExpirable()
            );

            Product honey = new Product(
                    "Honey",
                    5.0F,
                    8,
                    new Shippable(100,19.99F),
                    new NotExpirable()
            );

            Product clock = new Product(
                    "Clock",
                    5.0F,
                    100,
                    new NotShippable(),
                    new NotExpirable()
            );

            // Creating a Cart
            Cart cart = new Cart(4000);

            // Adding Items to cart
            cart.add(cheese, 15);
            cart.add(phone, 1);
            cart.add(honey, 4);
            cart.add(clock, 10);

            cart.checkout();

            System.out.println();
            // Printing all shippable items using ShippingService
            for (SimpleShippableI shippableItem :
ShippingService.toBeShipped){
                System.out.println("Shipping Item " + shippableItem.getName()
+ " with weight: " + shippableItem.getWeight());
            }


        } catch (Exception e) {
            System.out.println(e.getMessage());
        }

    }
}
```
Output:



```
Insufficient Balance
```

Code (Checking out an empty cart):

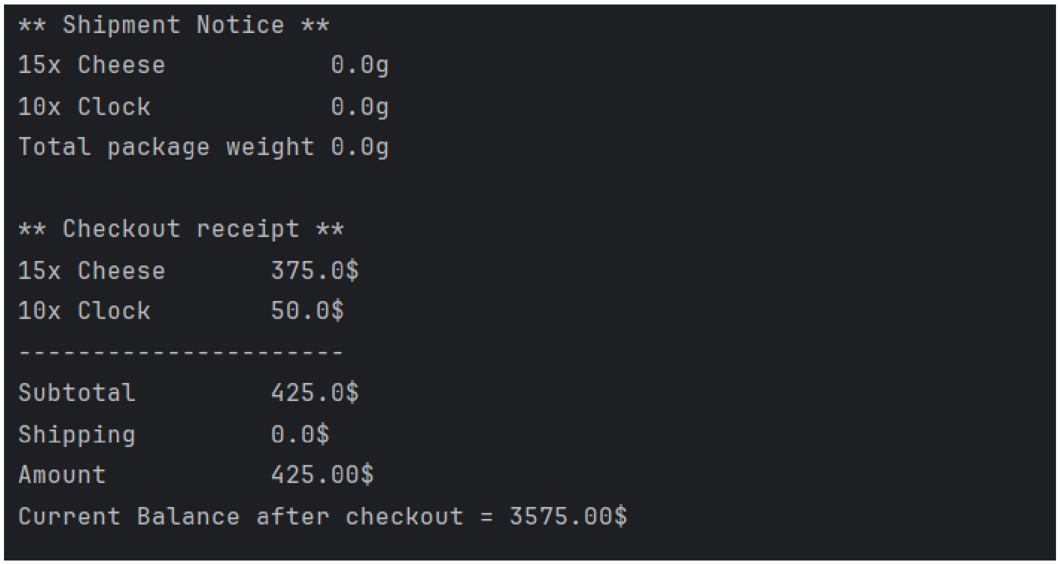
```
package org.example;

public class Main {
    public static void main(String[] args) {
        try {


            // Creating a Cart
            Cart cart = new Cart(4000);


            cart.checkout();

            System.out.println();
            // Printing all shippable items using ShippingService
            for (SimpleShippableI shippableItem :
ShippingService.toBeShipped){
                System.out.println("Shipping Item " + shippableItem.getName()
+ " with weight: " + shippableItem.getWeight());
            }


        } catch (Exception e) {
            System.out.println(e.getMessage());
        }

    }
}
```
Output:



```
Cart is Empty
```

```
Code (checking out with no shippable products):
package org.example;

public class Main {
    public static void main(String[] args) {
        try {
            Product cheese = new Product(
                    "Cheese",
                    25.0F,
                    15,
                    new NotShippable(),
                    new Expirable(2026)
            );

            Product clock = new Product(
                    "Clock",
                    5.0F,
                    100,
                    new NotShippable(),
                    new NotExpirable()
            );

            // Creating a Cart
            Cart cart = new Cart(4000);

            // Adding Items to cart
            cart.add(cheese, 15);
            cart.add(clock, 10);

            cart.checkout();

            System.out.println();
            // Printing all shippable items using ShippingService
            for (SimpleShippableI shippableItem :
ShippingService.toBeShipped){
                System.out.println("Shipping Item " + shippableItem.getName()
+ " with weight: " + shippableItem.getWeight());
            }


        } catch (Exception e) {
            System.out.println(e.getMessage());
        }

    }
}
```

Output:

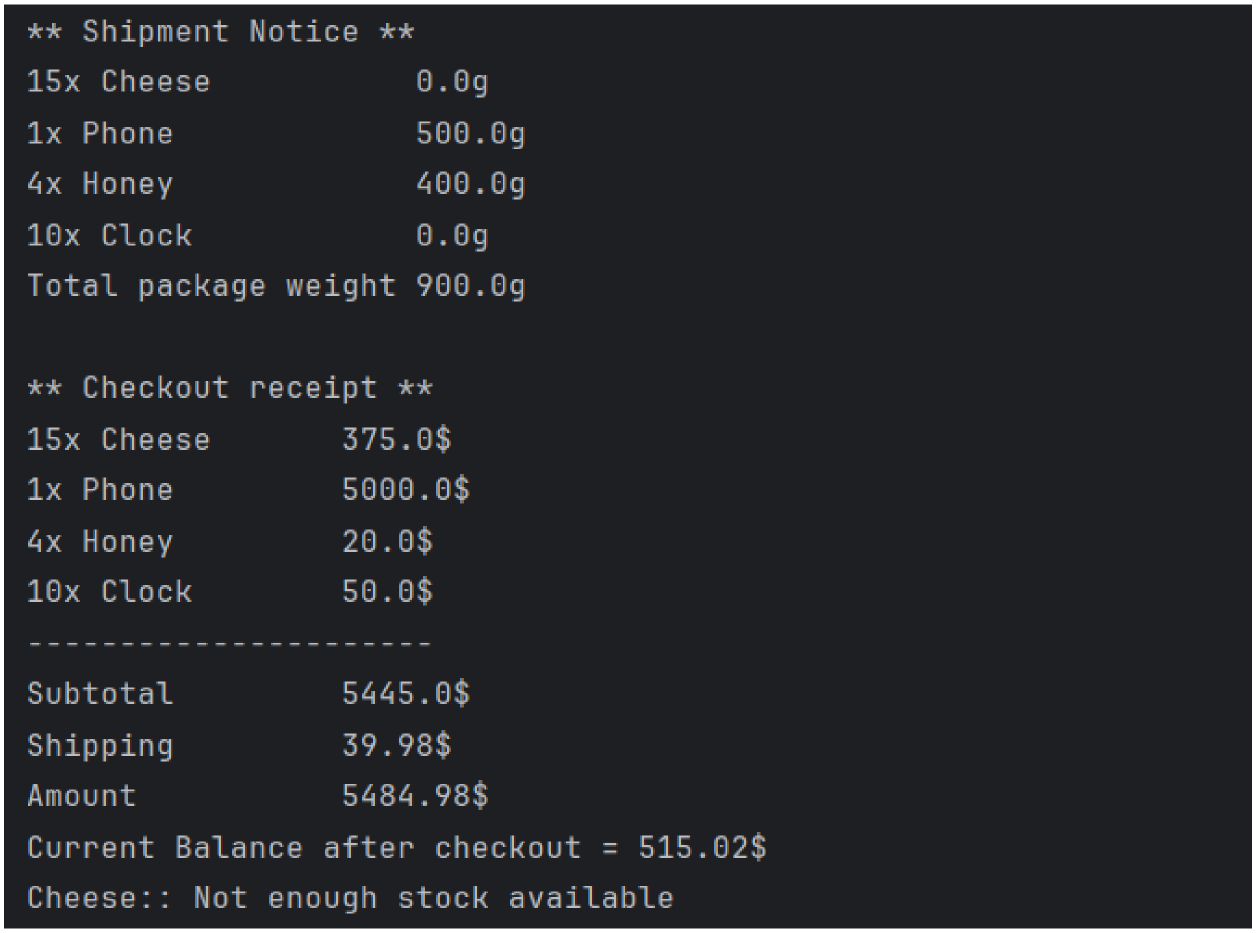


```
** Shipment Notice **
15x Cheese          0.0g
10x Clock           0.0g
Total package weight 0.0g

** Checkout receipt **
15x Cheese      375.0$
10x Clock       50.0$
----------------------
Subtotal        425.0$
Shipping        0.0$
Amount          425.00$
Current Balance after checkout = 3575.00$
```

Code (2 carts but not enough stock in cheese):

```java
package org.example;

public class Main {
    public static void main(String[] args) {
        try {
            Product cheese = new Product(
                    "Cheese",
                    25.0F,
                    15,
                    new NotShippable(),
                    new Expirable(2026)
            );

            Product phone = new Product(
                    "Phone",
                    5000.0F,
                    1,
                    new Shippable(500,19.99F),
                    new NotExpirable()
            );

            Product honey = new Product(
                    "Honey",
                    5.0F,
                    8,
                    new Shippable(100,19.99F),
                    new NotExpirable()
            );

            Product clock = new Product(
                    "Clock",
                    5.0F,
                    100,
                    new NotShippable(),
                    new NotExpirable()
            );

            // Creating a Cart
            Cart cart = new Cart(6000);

            // Adding Items to cart
            cart.add(cheese, 15);
            cart.add(phone, 1);
            cart.add(honey, 4);
            cart.add(clock, 10);

            cart.checkout();

            // Creating a Cart
            Cart cart1 = new Cart(2000);

            // Adding Items to cart
            cart1.add(cheese, 15);
            cart1.checkout();


            System.out.println();
            // Printing all shippable items using ShippingService
            for (SimpleShippableI shippableItem :
ShippingService.toBeShipped){
                System.out.println("Shipping Item " + shippableItem.getName()
+ " with weight: " + shippableItem.getWeight());
            }


        } catch (Exception e) {
            System.out.println(e.getMessage());
        }

    }
}
```

Output:



```
** Shipment Notice **
15x Cheese           0.0g
1x Phone             500.0g
4x Honey             400.0g
10x Clock            0.0g
Total package weight 900.0g

** Checkout receipt **
15x Cheese       375.0$
1x Phone         5000.0$
4x Honey         20.0$
10x Clock        50.0$
----------------------
Subtotal         5445.0$
Shipping         39.98$
Amount           5484.98$
Current Balance after checkout = 515.02$
Cheese:: Not enough stock available
```