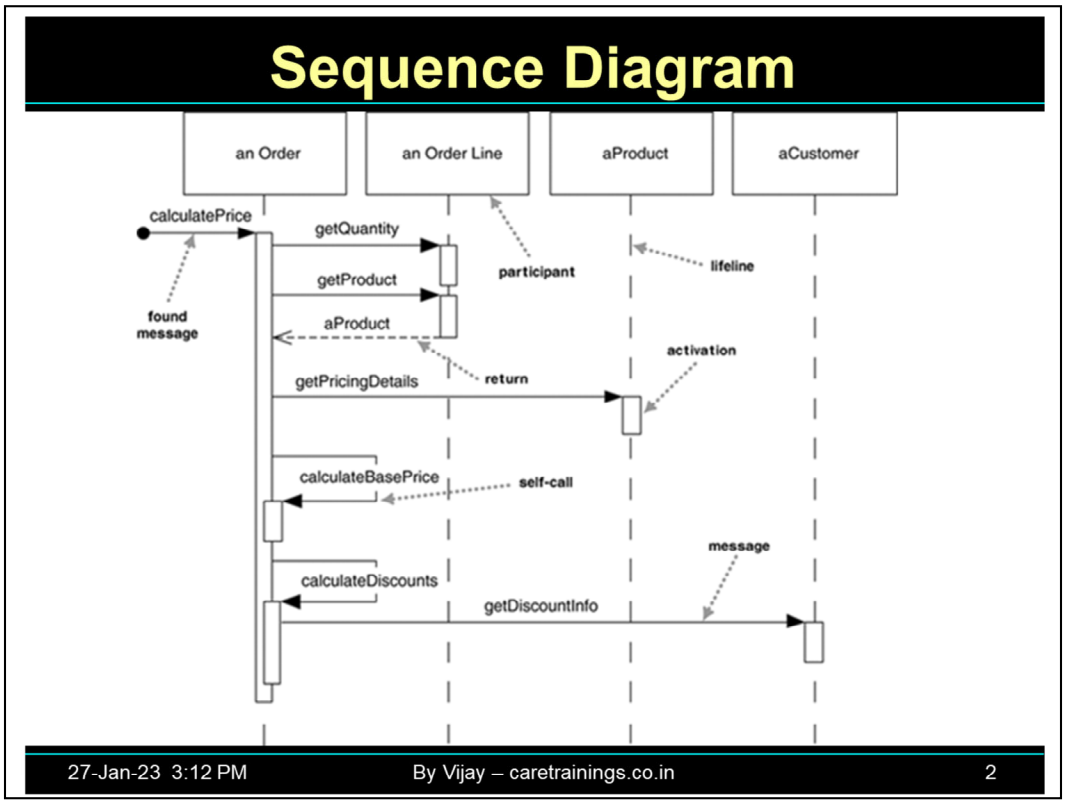
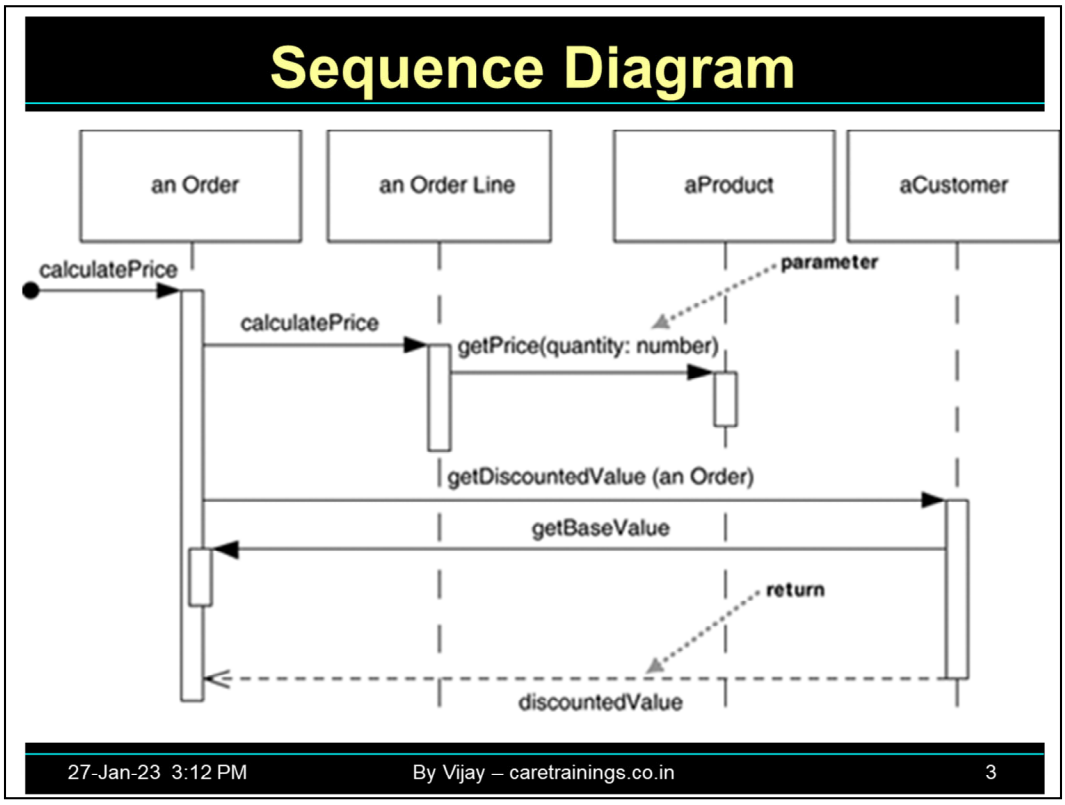
UML

Sequence Diagram




```
class Customer {
public:
        double getDiscountInfo() { return 0; }
};
class Product {
public:
        double getPricingDetails() { return 0; }
};
class OrderLine {
        Product* p;
        int quantity;
public:
        int getQuantity() { return quantity; }
        Product* getProduct() { return p; }
};
class Order {
        vector<OrderLine> lineItems;
```

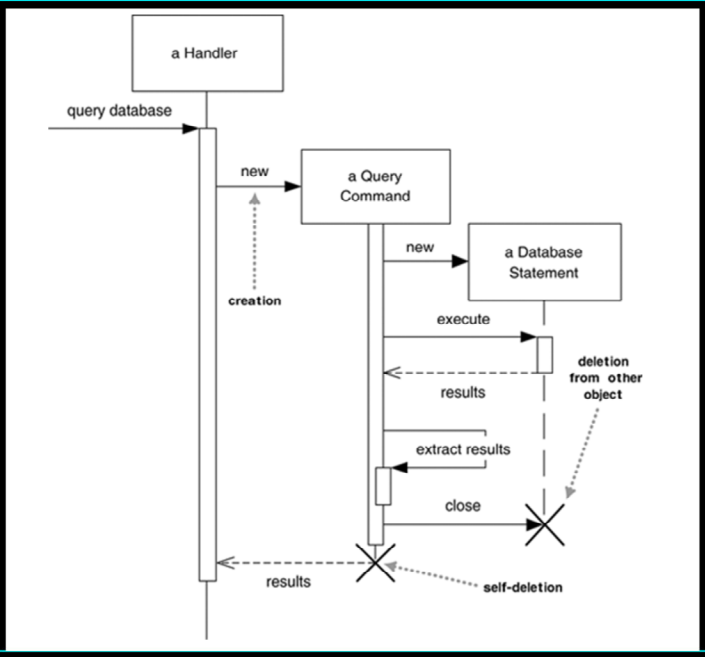
```
                double calculateBasePrice() { return 0; }
                double calculateDiscounts() { return 0; }
        public:
                double calculatePrice() {
                        double r = 0;
                        for (auto li : lineItems) {
                                auto q = li.getQuantity();
                                auto p = li.getProduct();
                                r += p->getPricingDetails();
                        }
                        return r + calculateBasePrice() +
calculateDiscounts();
                }
        };
```



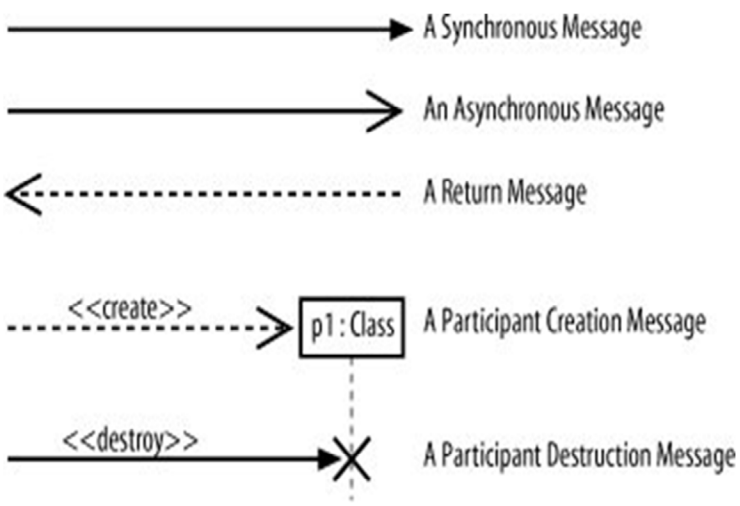

```
class Order;
class Customer {
public:
        double getDiscountedValue(Order* o);
};
class Product {
public:
        double getPrice(int quanity) { return 0; }
};
class OrderLine {
        Product* p;
        int quanity;
public:
        double calculatePrice() {
                return p->getPrice(quanity);
        }
};
```

```
class Order {
		Customer* c;
		vector<OrderLine> lineItems;
public:
		double getBaseValue() { return 0; }
		double calculatePrice() {
				double r = 0;
				for (auto li : lineItems)
						r += li.calculatePrice();
				return r - c->getDiscountedValue(this);
		}
};
double Customer::getDiscountedValue(Order* o) {
		double discountedValue = o->getBaseValue();
		//..
		return discountedValue;
}
```

Creation & Deletion



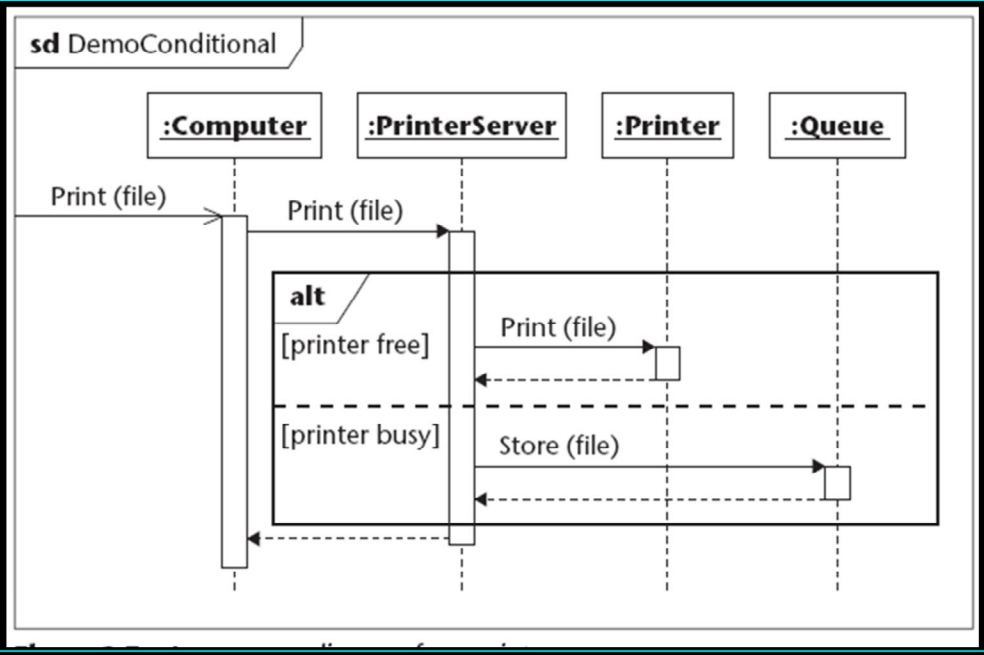
Message Arrows



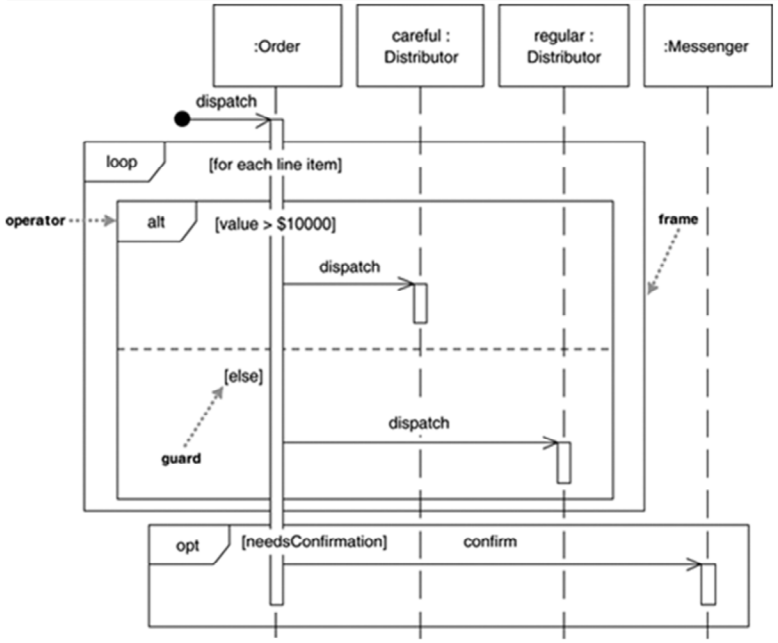
Message signatures

- doSomething( )
  - The message's name is doSomething, but no further information is known about it.
- doSomething(number1 : Number, number2 : Number)
  - The message's name is doSomething, and it takes two arguments, number1 and number2, which are both of class Number.
- doSomething( ) : ReturnClass
  - The message's name is doSomething; it takes no arguments and returns an object of class ReturnClass.
- myVar = doSomething( ) : ReturnClass
  - The message's name is doSomething; it takes no arguments, and it returns an object of class ReturnClass that is assigned to the myVar attribute of the message caller.

Print Server

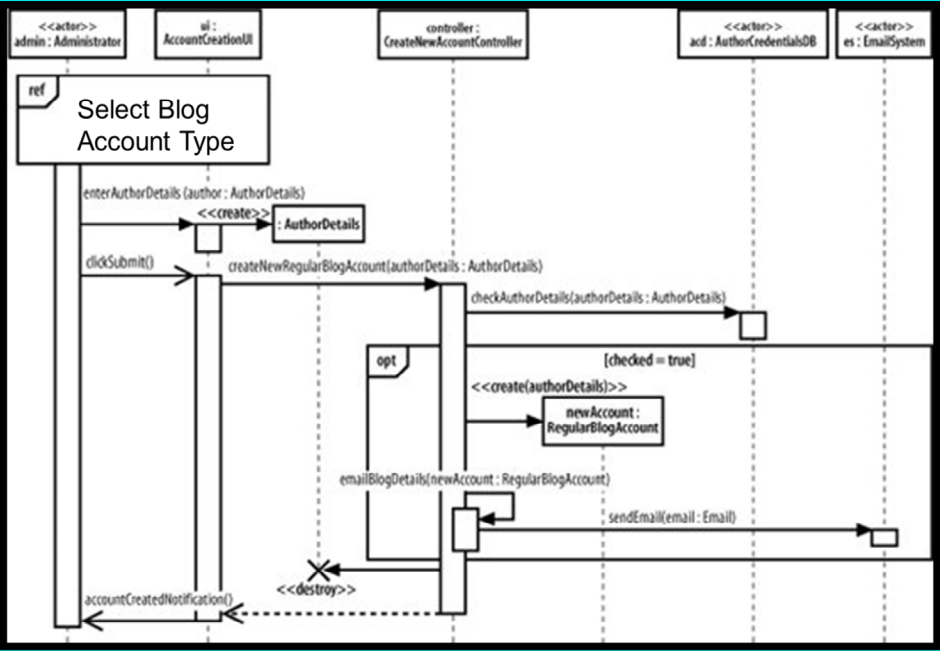


Loops & Conditionals

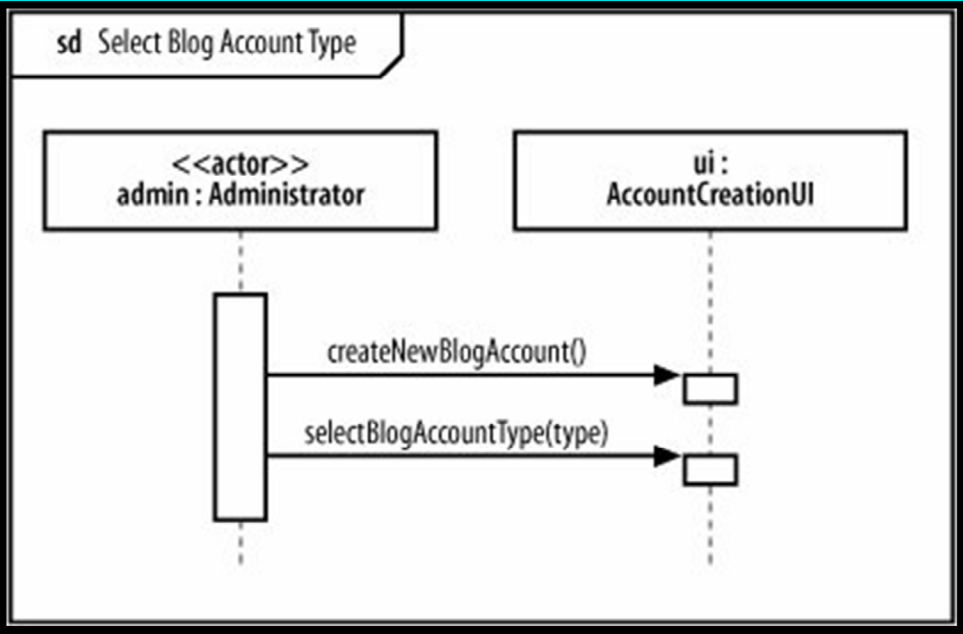



```
foreach (lineitem)
  if (product.value > $10K)
    careful.dispatch
  else
    regular.dispatch
  end if
end for
if (needsConfirmation)
  messenger.confirm
end procedure
```

Reference


<<actor>>
admin : Administrator
ui :
AccountCreationUI
controller :
CreateNewAccountController
<<actor>>
acd : AuthorCreedentialsDB
<<actor>>
es : EmailSystem
ref
Select Blog
Account Type
enterAuthorDetails (author : AuthorDetails)
<<create>>
: AuthorDetails
clickSubmit()
createNewRegularBlogAccount(authorDetails : AuthorDetails)
checkAuthorDetails(authorDetails : AuthorDetails)
opt
[checked = true]
<<create(authorDetails)>>
newAccount :
RegularBlogAccount
emailBlogDetails(newAccount : RegularBlogAccount)
sendEmail(email : Email)
<<destroy>>
accountCreatedNotification()

Reference Details



Other operators for Frames

- par - Parallel; each fragment is run in parallel.
- region - Critical region; the fragment can have only one thread executing it at once.
- neg - Negative; the fragment shows an invalid interaction.