ESOF 322: Homework 2

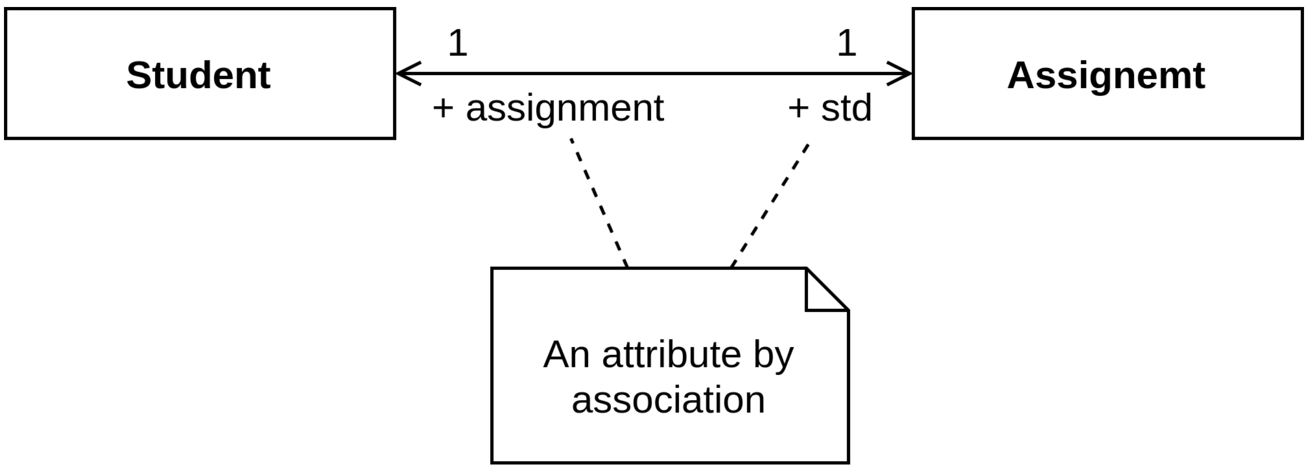
River Kelly

September 21, 2021

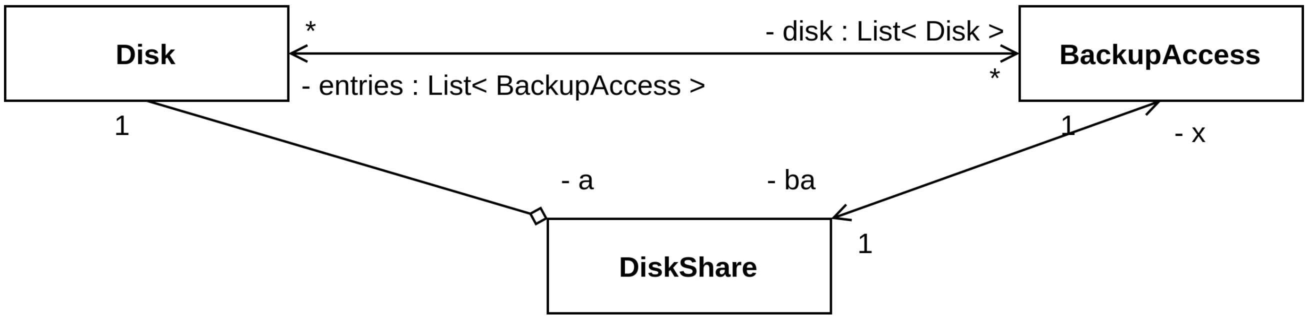
Partner: Peyton Dorsh

Exercise Part A (15 pts)

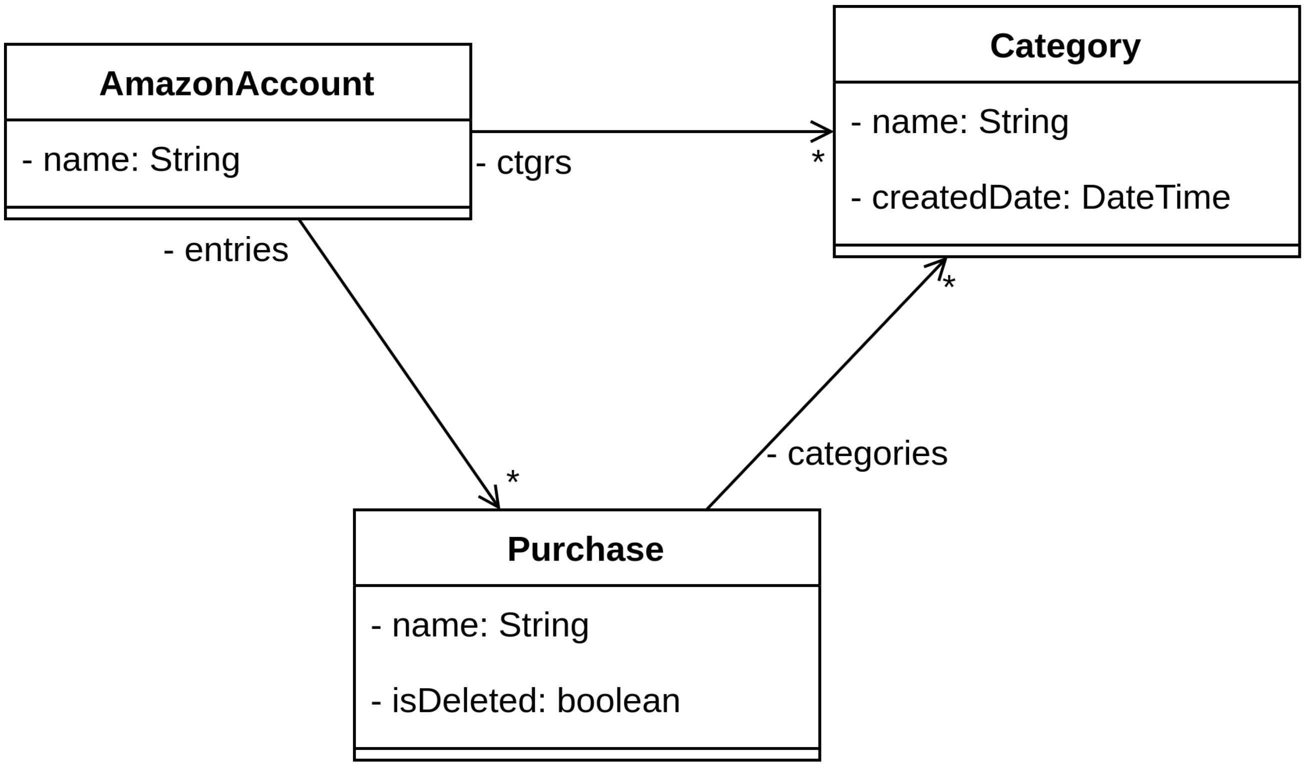
1. (2pts)



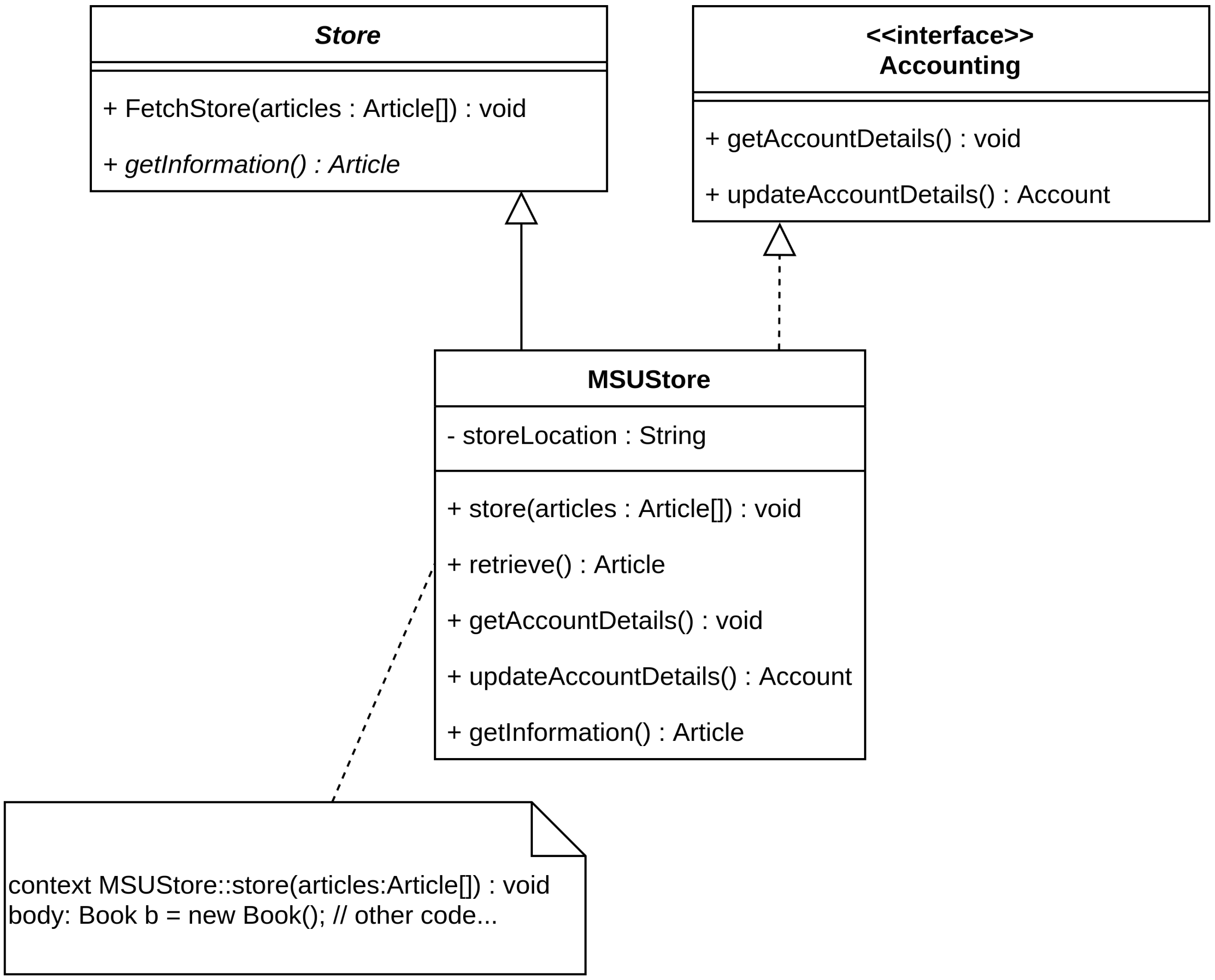
2. (3pts)



3. (5pts)


AmazonAccount
- name: String
Category
- name: String
- createdDate: DateTime
Purchase
- name: String
- isDeleted: boolean
- ctgrs
*
- entries
*
- categories
*

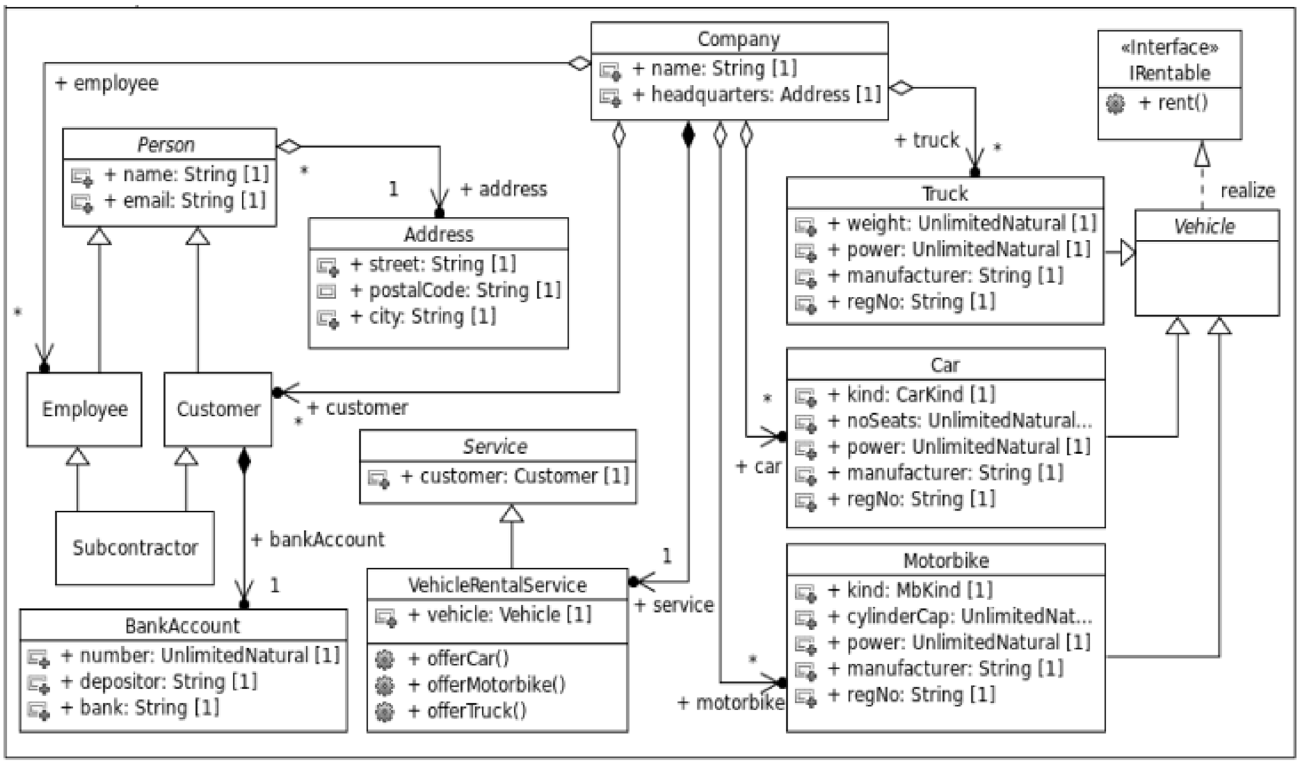
4. (5pts)



Store
+ FetchStore(articles : Article[]) : void
+ getInformation() : Article
<<interface>>
Accounting
+ getAccountDetails() : void
+ updateAccountDetails() : Account
MSUStore
- storeLocation : String
+ store(articles : Article[]) : void
+ retrieve() : Article
+ getAccountDetails() : void
+ updateAccountDetails() : Account
+ getInformation() : Article
context MSUStore::store(articles:Article[]) : void
body: Book b = new Book(); // other code...

Exercise Part B (15 pts)

Write **pseudo code** to describe the following UML class diagram:



Company Class

```
public class Company {
  public String name;
  public Address headquarters;

  // properties from associations
  public Customer customer;
  public Employee employee;
  public VehicleRentalService service;
  public Truck truck;
  public Car car;
  public Motorbike motorbike;

  // Company Destructor
  public void finalize() {
    delete service (this.service)
    delete self (delete this)
  }
}
```

Service Class

```
public abstract class Service {
    public Customer customer;
}
```

VehicleRentalService Class

```
public abstract class VehicleRentalService {
    public Vehicle vehicle;
    public offerCar();
    public offerMotorbike();
    public offerTruck();
}
```

IRentable Class

```
public interface IRentable {
  public void rent() {}
}
```

Vehicle Class

```
public class Vehicle implements IRentable {
  public UnlimitedNatural power;
  public String manufacturer;
  public String regNo;

  public void rent() {
    some code to rent the vehicle
  }
}
```

Truck Class

```
public class Truck extends Vehicle {
  public UnlimitedNatural weight;
}
```

Car Class

```
public class Car extends Vehicle {
  public CarKind kind;
  public UnlimitedNatural noSeats;
}
```

Motorbike Class

```
public class Motorbike extends Vehicle {
  public MbKind kind;
  public UnlimitedNatural cylinderCap;
}
```

Person Class

```
public class Person {
  public String name;
  public String email;
  // properties from associations
  public Address address;
}
```

Address Class

```java
public class Address {
  public String street;
  public String postalCode;
  public String city;
}
```

Customer Class

```java
public class Customer extends Person {

  // properties from associations
  public BankAccount bankAccount;

  // Customer Destructor
  public void finalize() {
    delete bank account (this.bankAccount)
    delete self (delete this)
  }

}
```

Employee Class

```java
public class Employee extends Person {}
```

Subcontractor Class

```java
public class Subcontractor extends Person, Employee {}
```

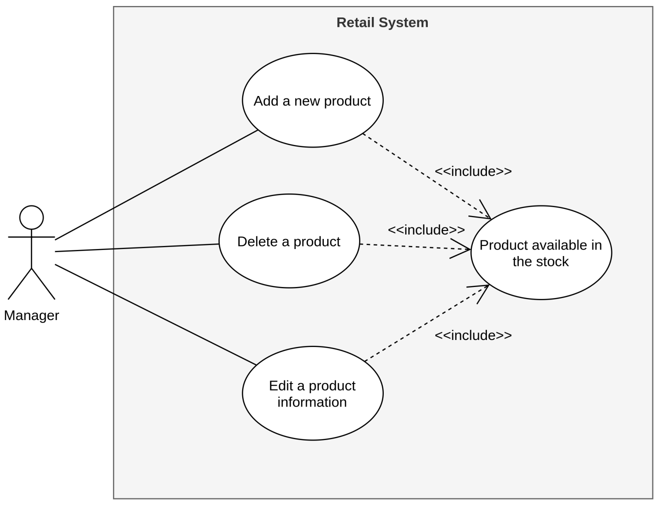
BankAccount Class

```java
public class BankAccount {
  public UnlimitedNatural number;
  public String depositor;
  public String bank;
}
```

Exercise Part C (5 pts)

Suppose we need to develop a system named 'Retail System'. Draw a single use case diagram capturing the following 4 use cases.

- A manager can add a new product in the system.
- A manager can delete a product in the system.
- A manager can edit a product information in the system.
- Both use cases (i), (ii) and (iii) should reuse this new use case i.e., a product should be available in the stock.



Use Case: Add a new Product

Use case name:	Add a new Product
Goal In Content:	A Manager requests to create and add a new product to the store inventory
Preconditions:	Check if a duplicate product already exists in the Store inventory
Successful End Condition:	A new product is added to the store inventory
Failed End Condition:	The request for creating a new Product in the store's inventory is rejected
Primary Actors:	Manager
Trigger:	The Manager asks the Retail Store to create a new product
Main Flow:	1. The Manager requests the Retail Store to create a new product 2. See if product is already available (i.e. if it is a duplicate) 3. The new product is created 4. The product is available in the store's inventory

Use Case: Delete a Product in the system

Use case name:	Delete a product in the system
Goal In Content:	A Manager requests to remove an existing product from the store's inventory
Preconditions:	The product must exist in the store's inventory
Successful End Condition:	The product is deleted from the store's inventory
Failed End Condition:	The product is not removed from the store's inventory
Primary Actors:	Manager
Trigger:	The Manager asks the Retail Store to delete a product
Main Flow:	1. The Manager makes a request to the Retail Store to remove a product 2. The systems checks if the product exists in the store's inventory 3. The product is removed from the store's inventory

Use Case: Edit product information

Use case name:	Edit a product's information in the system
Goal In Content:	A Manager requests to update the information of an existing product in the store's inventory
Preconditions:	The product must exist in the store's inventory
Successful End Condition:	The product's information is updated in the system
Failed End Condition:	The product's information is not updated
Primary Actors:	Manager
Trigger:	The Manager asks the Retail Store to update a product's information
Main Flow:	1. The Manager makes a request to the Retail Store to update a product's information 2. The systems checks if the product exists in the store's inventory 3. The product's information is updated in the system

Use Case: A product should be available in the stock

Use case name:	A product should be available in the stock
Goal In Content:	The existence of a product in the system is checked
Successful End Condition:	The product is in stock
Failed End Condition:	The product is not in stock
Primary Actors:	Manager
Trigger:	A request to update an item in the store's inventory is made
Main Flow:	1. A Manager requests to make some update to the store's inventory 2. The systems checks if the product is in stock