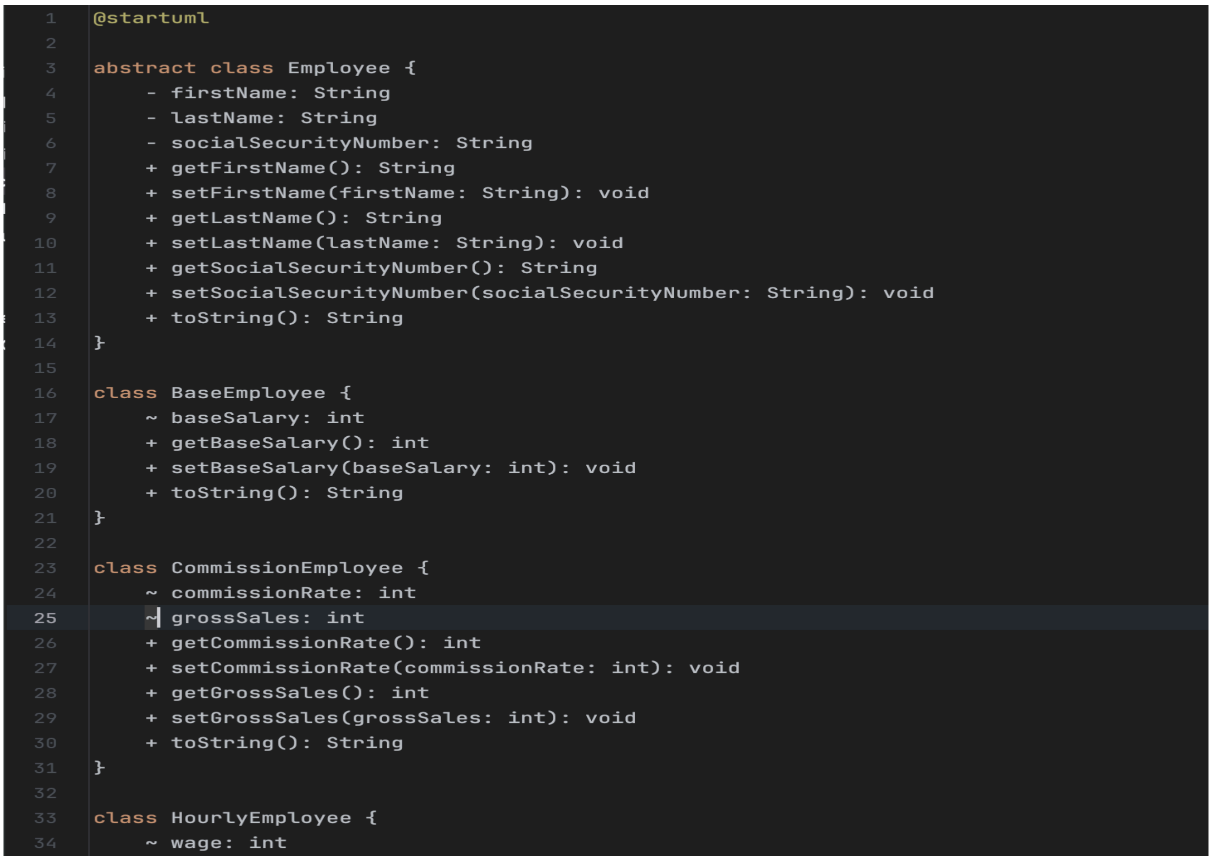
CS 5800.01 - Advanced Software Engineering

Homework-3

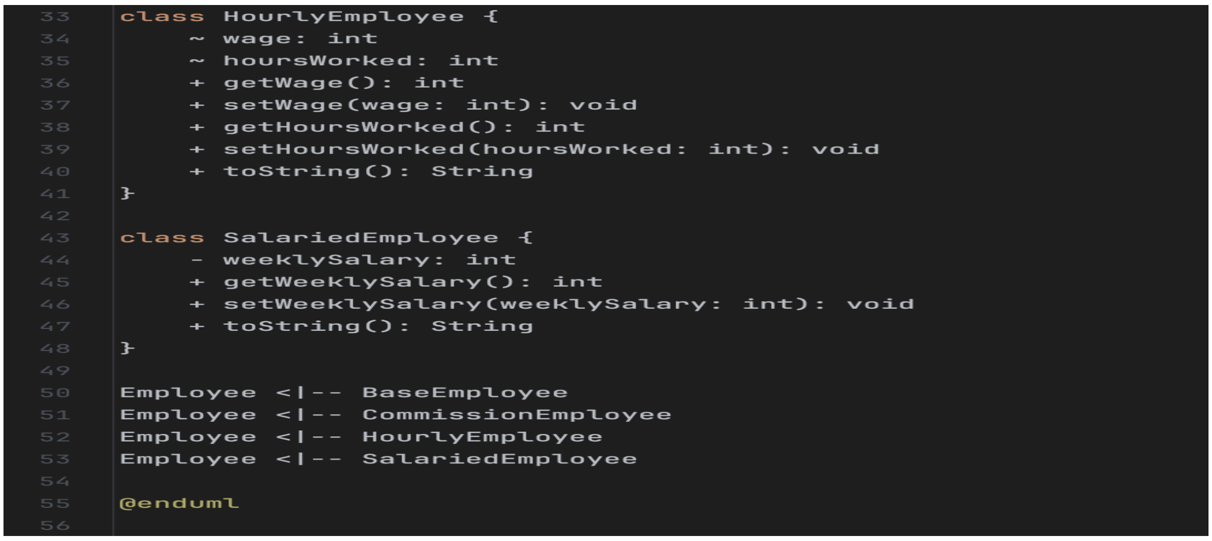
Answers

Github link: https://github.com/shreyas463/AdvEng-Assignment3.git

1- Inheritance (Code, Class Diagram)

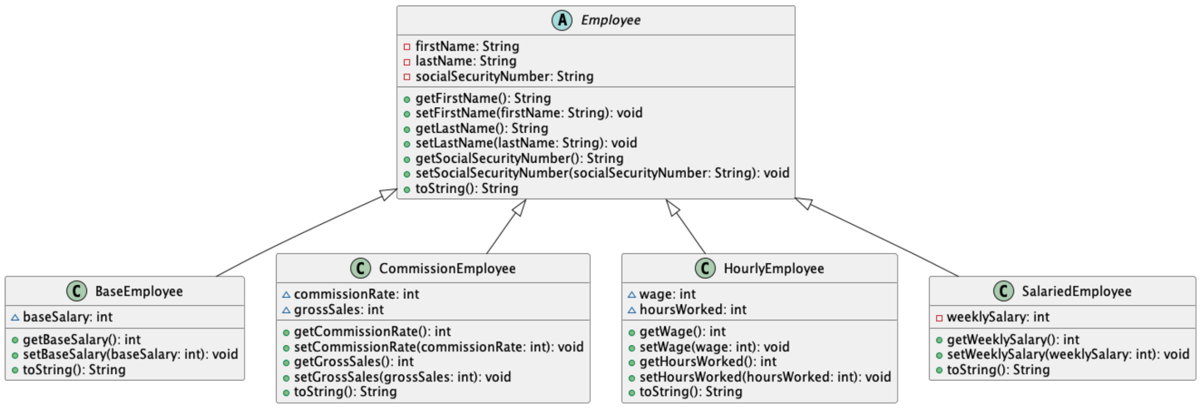


```
@startuml

abstract class Employee {
    - firstName: String
    - lastName: String
    - socialSecurityNumber: String
    + getFirstName(): String
    + setFirstName(firstName: String): void
    + getLastName(): String
    + setLastName(lastName: String): void
    + getSocialSecurityNumber(): String
    + setSocialSecurityNumber(socialSecurityNumber: String): void
    + toString(): String
}

class BaseEmployee {
    ~ baseSalary: int
    + getBaseSalary(): int
    + setBaseSalary(baseSalary: int): void
    + toString(): String
}

class CommissionEmployee {
    ~ commissionRate: int
    ~ grossSales: int
    + getCommissionRate(): int
    + setCommissionRate(commissionRate: int): void
    + getGrossSales(): int
    + setGrossSales(grossSales: int): void
    + toString(): String
}

class HourlyEmployee {
    ~ wage: int
```



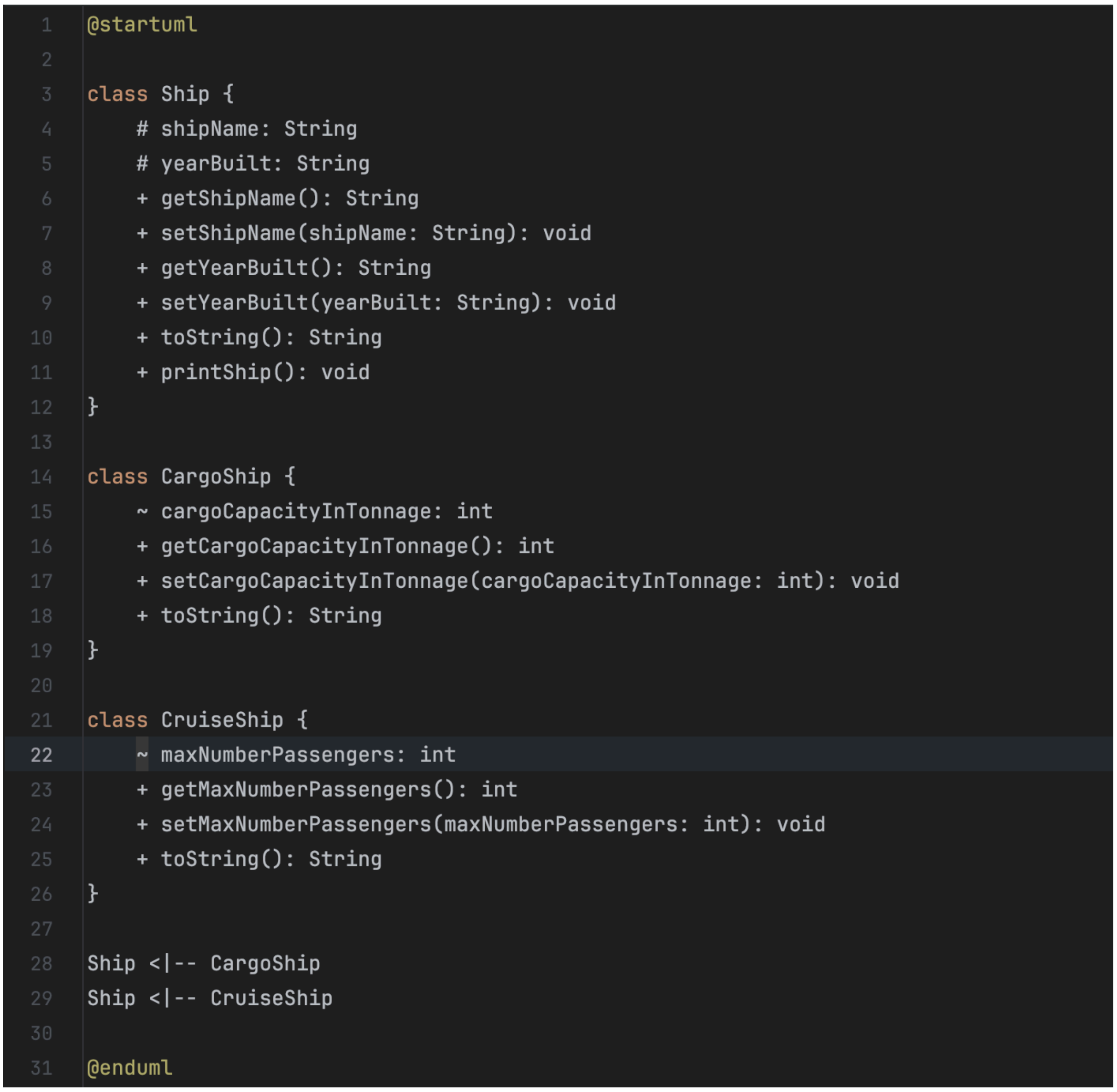
```
class HourlyEmployee {
    ~ wage: int
    ~ hoursWorked: int
    + getWage(): int
    + setWage(wage: int): void
    + getHoursWorked(): int
    + setHoursWorked(hoursWorked: int): void
    + toString(): String
}

class SalariedEmployee {
    - weeklySalary: int
    + getWeeklySalary(): int
    + setWeeklySalary(weeklySalary: int): void
    + toString(): String
}

Employee <|-- BaseEmployee
Employee <|-- CommissionEmployee
Employee <|-- HourlyEmployee
Employee <|-- SalariedEmployee

@enduml
```

Class Diagram Output: Inheritance



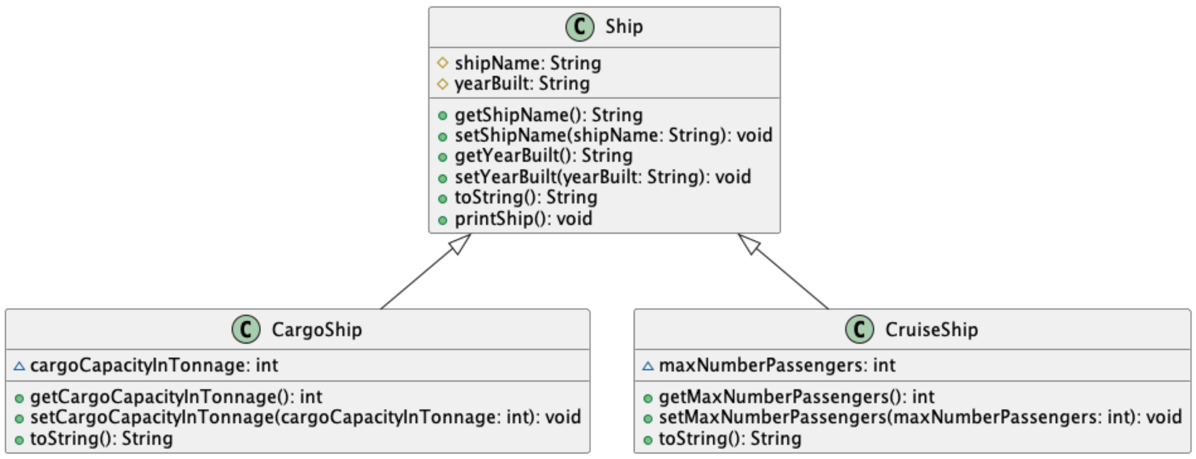
2) **Polymorphism (Code, class diagram, object diagram)**

Code for the class diagram:

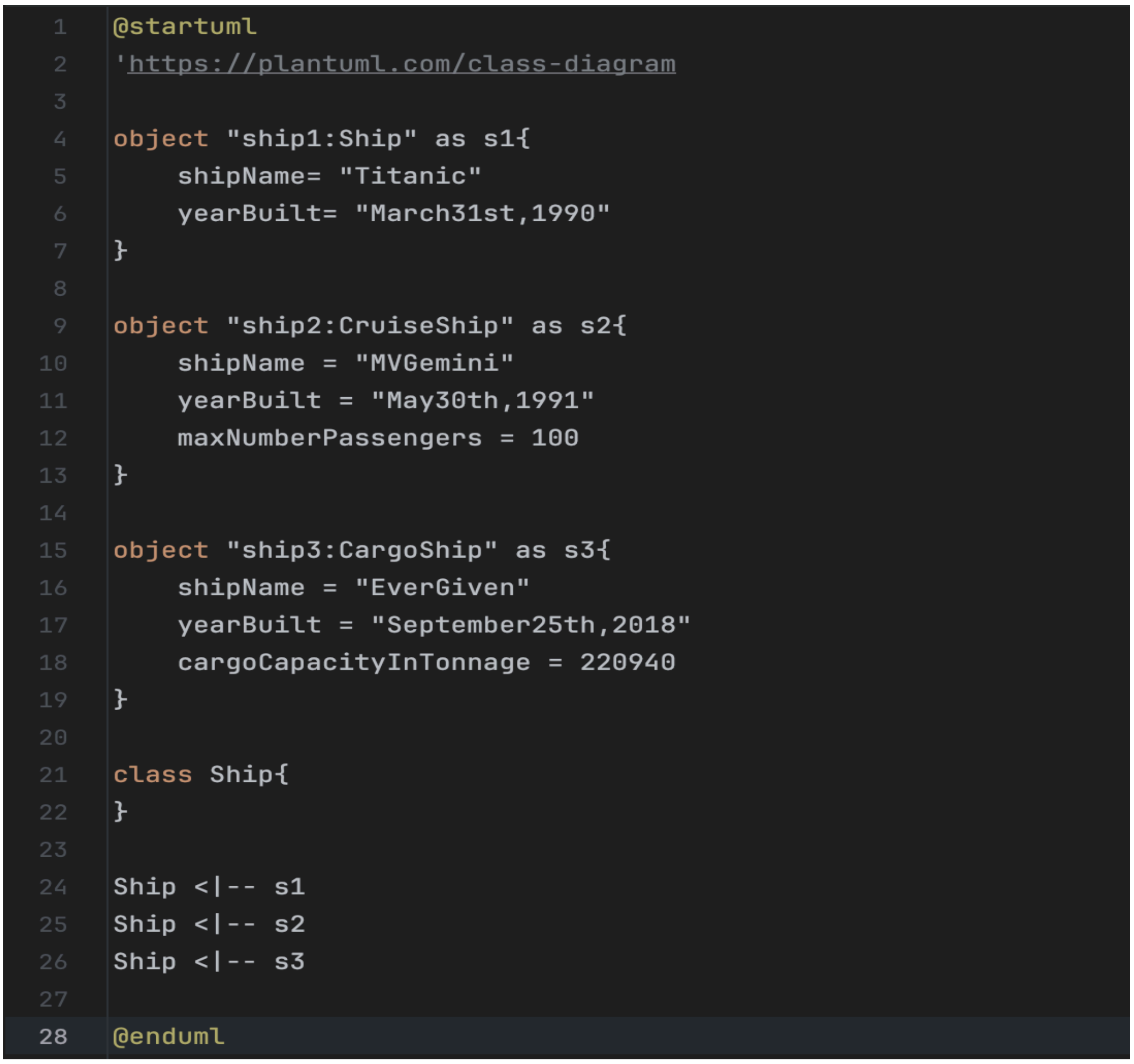


```
@startuml

class Ship {
    # shipName: String
    # yearBuilt: String
    + getShipName(): String
    + setShipName(shipName: String): void
    + getYearBuilt(): String
    + setYearBuilt(yearBuilt: String): void
    + toString(): String
    + printShip(): void
}

class CargoShip {
    ~ cargoCapacityInTonnage: int
    + getCargoCapacityInTonnage(): int
    + setCargoCapacityInTonnage(cargoCapacityInTonnage: int): void
    + toString(): String
}

class CruiseShip {
    ~ maxNumberPassengers: int
    + getMaxNumberPassengers(): int
    + setMaxNumberPassengers(maxNumberPassengers: int): void
    + toString(): String
}

Ship <|-- CargoShip
Ship <|-- CruiseShip

@enduml
```

2a: Class Diagram - Polymorphism

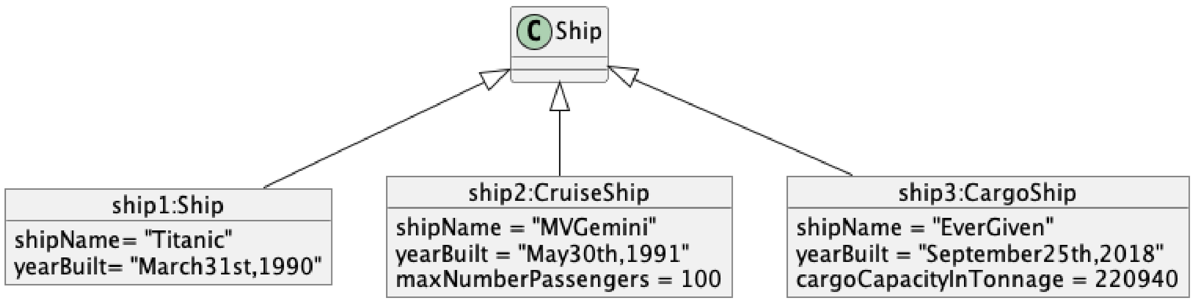

Ship
shipName: String
yearBuilt: String
getShipName(): String
setShipName(shipName: String): void
getYearBuilt(): String
setYearBuilt(yearBuilt: String): void
toString(): String
printShip(): void
CargoShip
cargoCapacityInTonnage: int
getCargoCapacityInTonnage(): int
setCargoCapacityInTonnage(cargoCapacityInTonnage: int): void
toString(): String
CruiseShip
maxNumberPassengers: int
getMaxNumberPassengers(): int
setMaxNumberPassengers(maxNumberPassengers: int): void
toString(): String

2b: Code for the Object Diagram

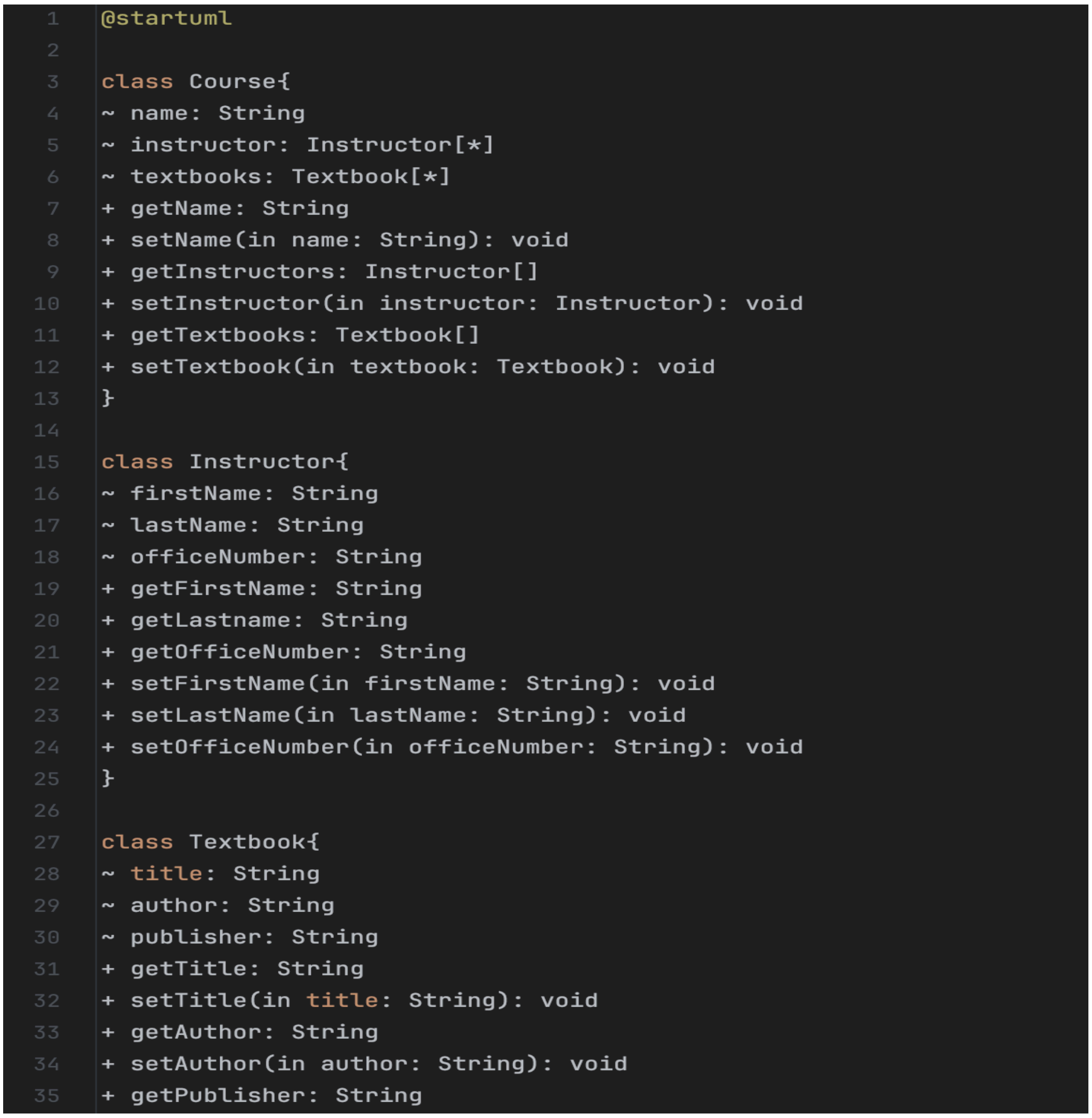


```
@startuml
'https://plantuml.com/class-diagram

object "ship1:Ship" as s1{
    shipName= "Titanic"
    yearBuilt= "March31st,1990"
}

object "ship2:CruiseShip" as s2{
    shipName = "MVGemini"
    yearBuilt = "May30th,1991"
    maxNumberPassengers = 100
}

object "ship3:CargoShip" as s3{
    shipName = "EverGiven"
    yearBuilt = "September25th,2018"
    cargoCapacityInTonnage = 220940
}

class Ship{
}

Ship <|-- s1
Ship <|-- s2
Ship <|-- s3

@enduml
```

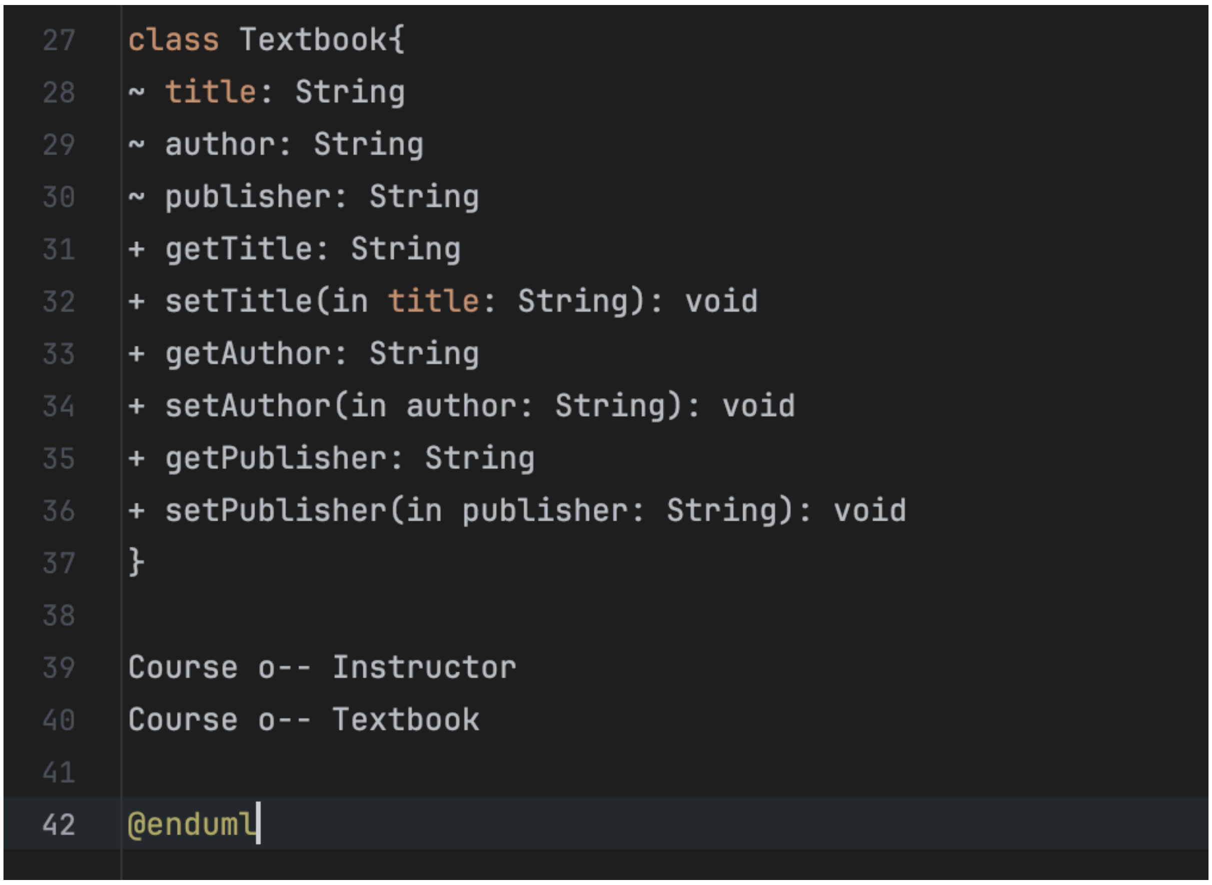
2b: Object Diagram - Polymorphism


Ship
ship1:Ship
shipName= "Titanic"
yearBuilt= "March31st,1990"
ship2:CruiseShip
shipName = "MVGemini"
yearBuilt = "May30th,1991"
maxNumberPassengers = 100
ship3:CargoShip
shipName = "EverGiven"
yearBuilt = "September25th,2018"
cargoCapacityInTonnage = 220940

3) Aggregation (Code, Class Diagram, Object Diagram)
Code for the class diagram:

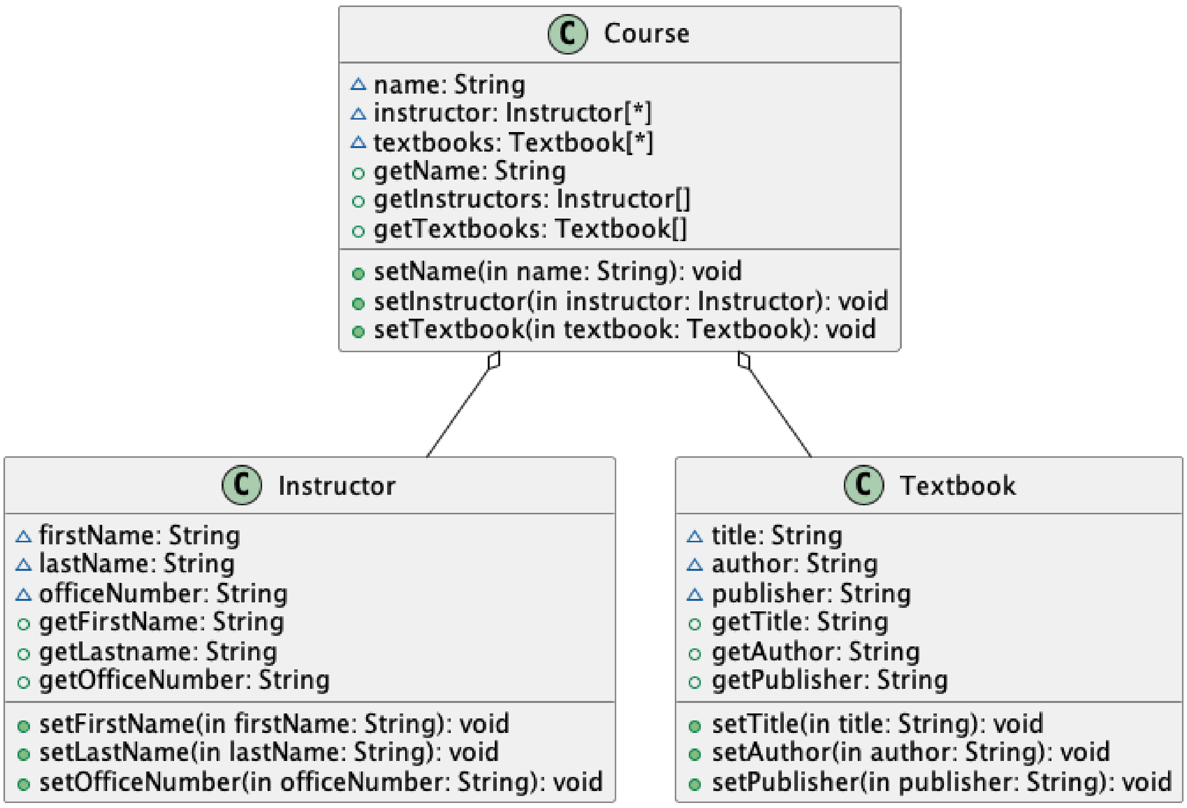


```
@startuml

class Course{
~ name: String
~ instructor: Instructor[*]
~ textbooks: Textbook[*]
+ getName: String
+ setName(in name: String): void
+ getInstructors: Instructor[]
+ setInstructor(in instructor: Instructor): void
+ getTextbooks: Textbook[]
+ setTextbook(in textbook: Textbook): void
}

class Instructor{
~ firstName: String
~ lastName: String
~ officeNumber: String
+ getFirstName: String
+ getLastname: String
+ getOfficeNumber: String
+ setFirstName(in firstName: String): void
+ setLastName(in lastName: String): void
+ setOfficeNumber(in officeNumber: String): void
}

class Textbook{
~ title: String
~ author: String
~ publisher: String
+ getTitle: String
+ setTitle(in title: String): void
+ getAuthor: String
+ setAuthor(in author: String): void
+ getPublisher: String
```

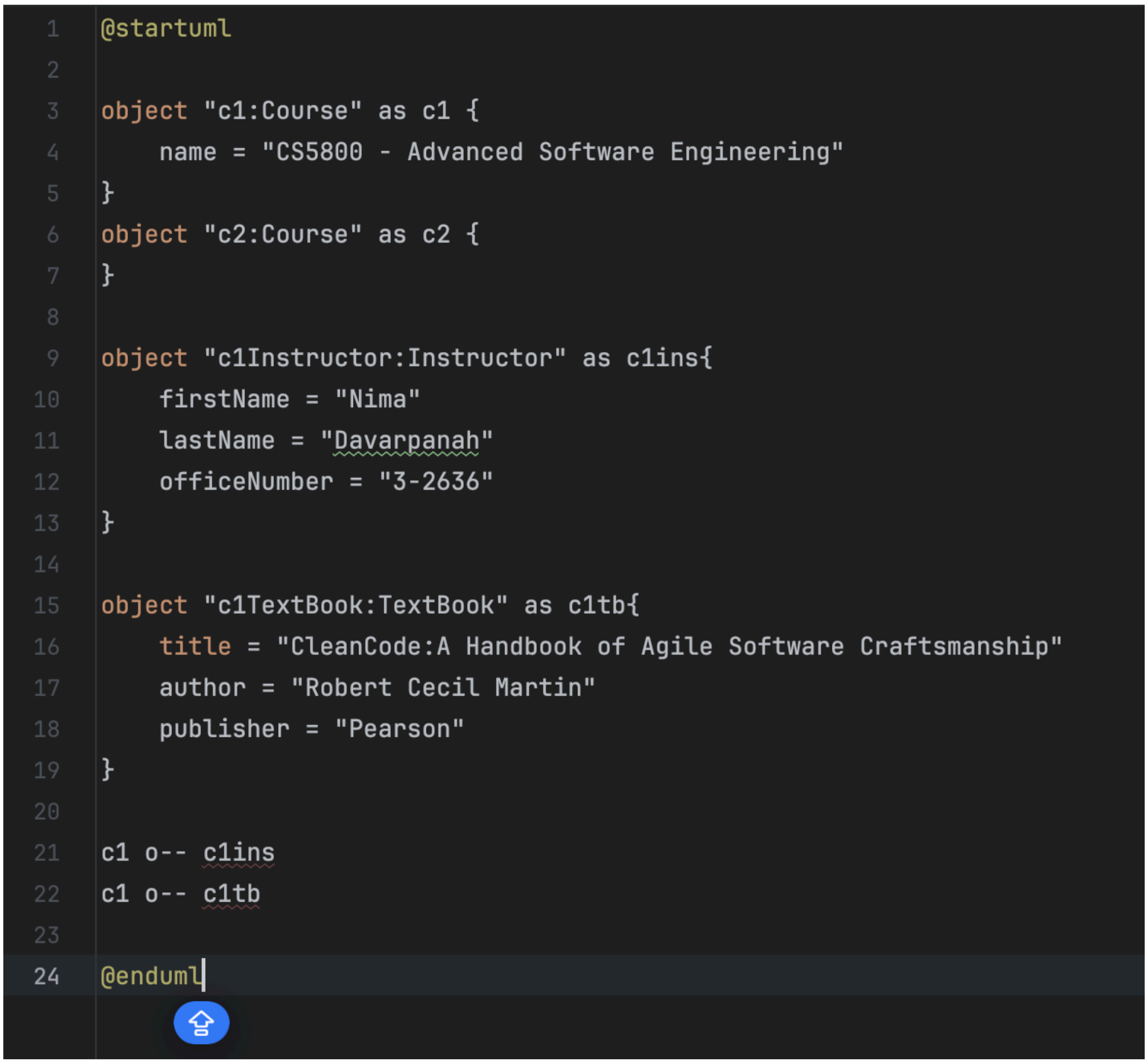


```
class Textbook{
~ title: String
~ author: String
~ publisher: String
+ getTitle: String
+ setTitle(in title: String): void
+ getAuthor: String
+ setAuthor(in author: String): void
+ getPublisher: String
+ setPublisher(in publisher: String): void
}

Course o-- Instructor
Course o-- Textbook

@enduml
```

3a: Class Diagram - Aggregation

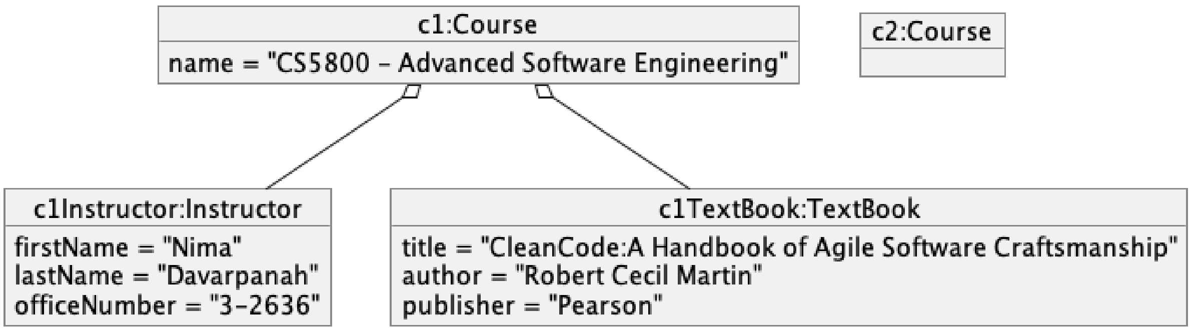

C Course
△ name: String
△ instructor: Instructor[*]
△ textbooks: Textbook[*]
○ getName: String
○ getInstructors: Instructor[]
○ getTextbooks: Textbook[]
● setName(in name: String): void
● setInstructor(in instructor: Instructor): void
● setTextbook(in textbook: Textbook): void
C Instructor
△ firstName: String
△ lastName: String
△ officeNumber: String
○ getFirstName: String
○ getLastname: String
○ getOfficeNumber: String
● setFirstName(in firstName: String): void
● setLastName(in lastName: String): void
● setOfficeNumber(in officeNumber: String): void
C Textbook
△ title: String
△ author: String
△ publisher: String
○ getTitle: String
○ getAuthor: String
○ getPublisher: String
● setTitle(in title: String): void
● setAuthor(in author: String): void
● setPublisher(in publisher: String): void

3b: Code for the Object Diagram



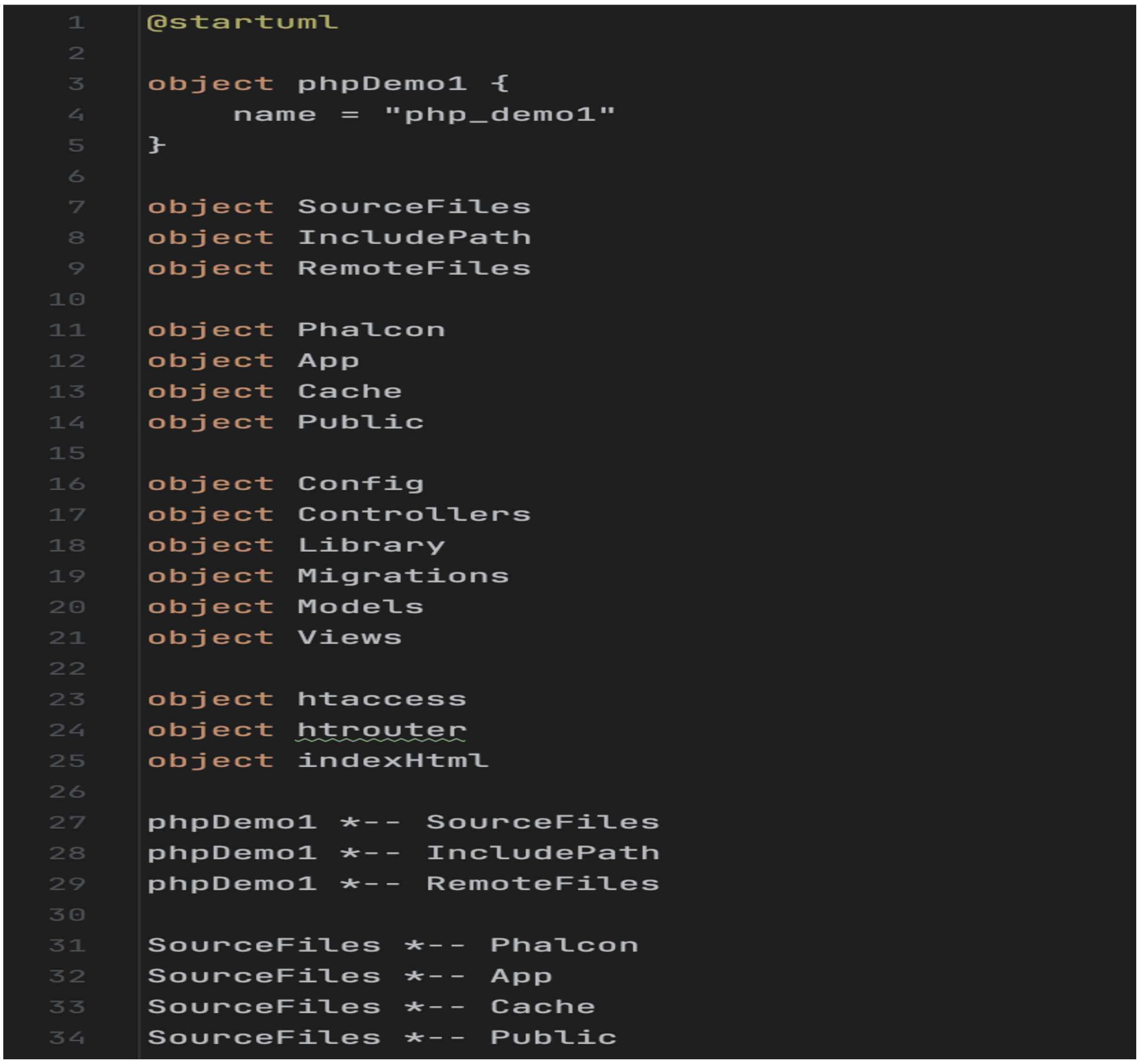
```
@startuml

object "c1:Course" as c1 {
    name = "CS5800 - Advanced Software Engineering"
}
object "c2:Course" as c2 {
}

object "c1Instructor:Instructor" as c1ins{
    firstName = "Nima"
    lastName = "Davarpanah"
    officeNumber = "3-2636"
}

object "c1TextBook:TextBook" as c1tb{
    title = "CleanCode:A Handbook of Agile Software Craftsmanship"
    author = "Robert Cecil Martin"
    publisher = "Pearson"
}

c1 o-- c1ins
c1 o-- c1tb

@enduml
```

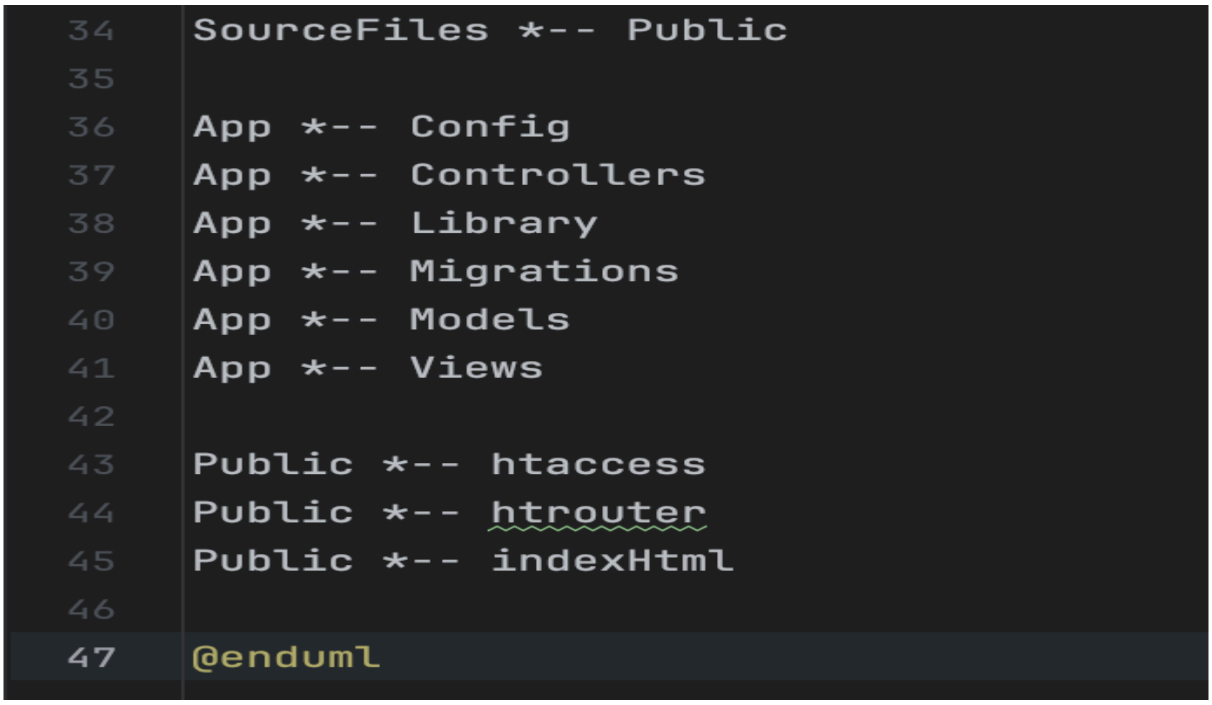
3b: Object Diagram - Aggregation



4) Composition (Code, Class Diagram, Object Diagram)

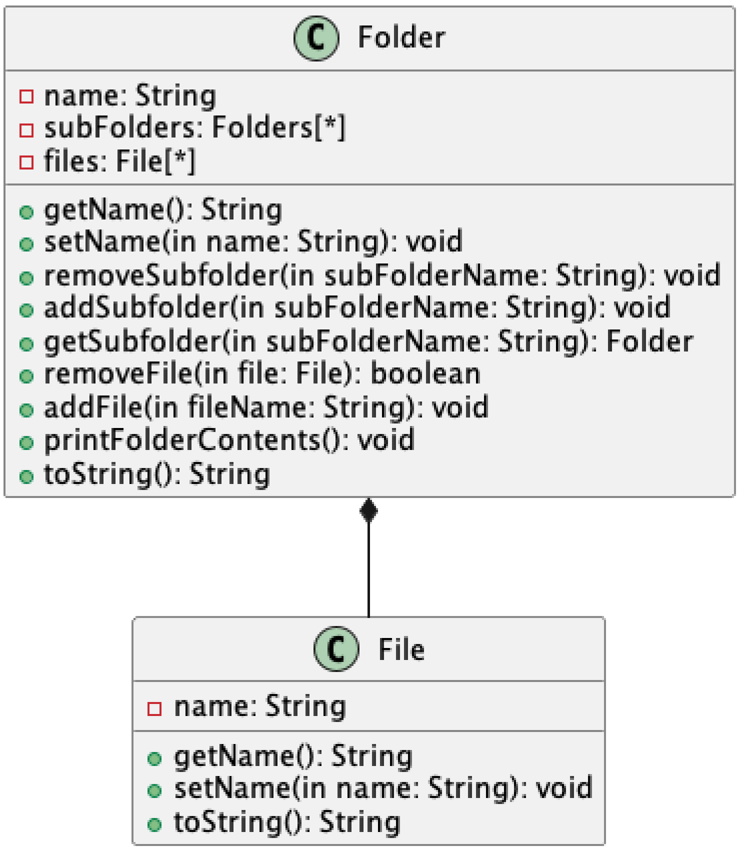
Code for the class diagram:


```
@startuml

object phpDemo1 {
    name = "php_demo1"
}

object SourceFiles
object IncludePath
object RemoteFiles

object Phalcon
object App
object Cache
object Public

object Config
object Controllers
object Library
object Migrations
object Models
object Views

object htaccess
object htrouter
object indexHtml

phpDemo1 *-- SourceFiles
phpDemo1 *-- IncludePath
phpDemo1 *-- RemoteFiles

SourceFiles *-- Phalcon
SourceFiles *-- App
SourceFiles *-- Cache
SourceFiles *-- Public
```

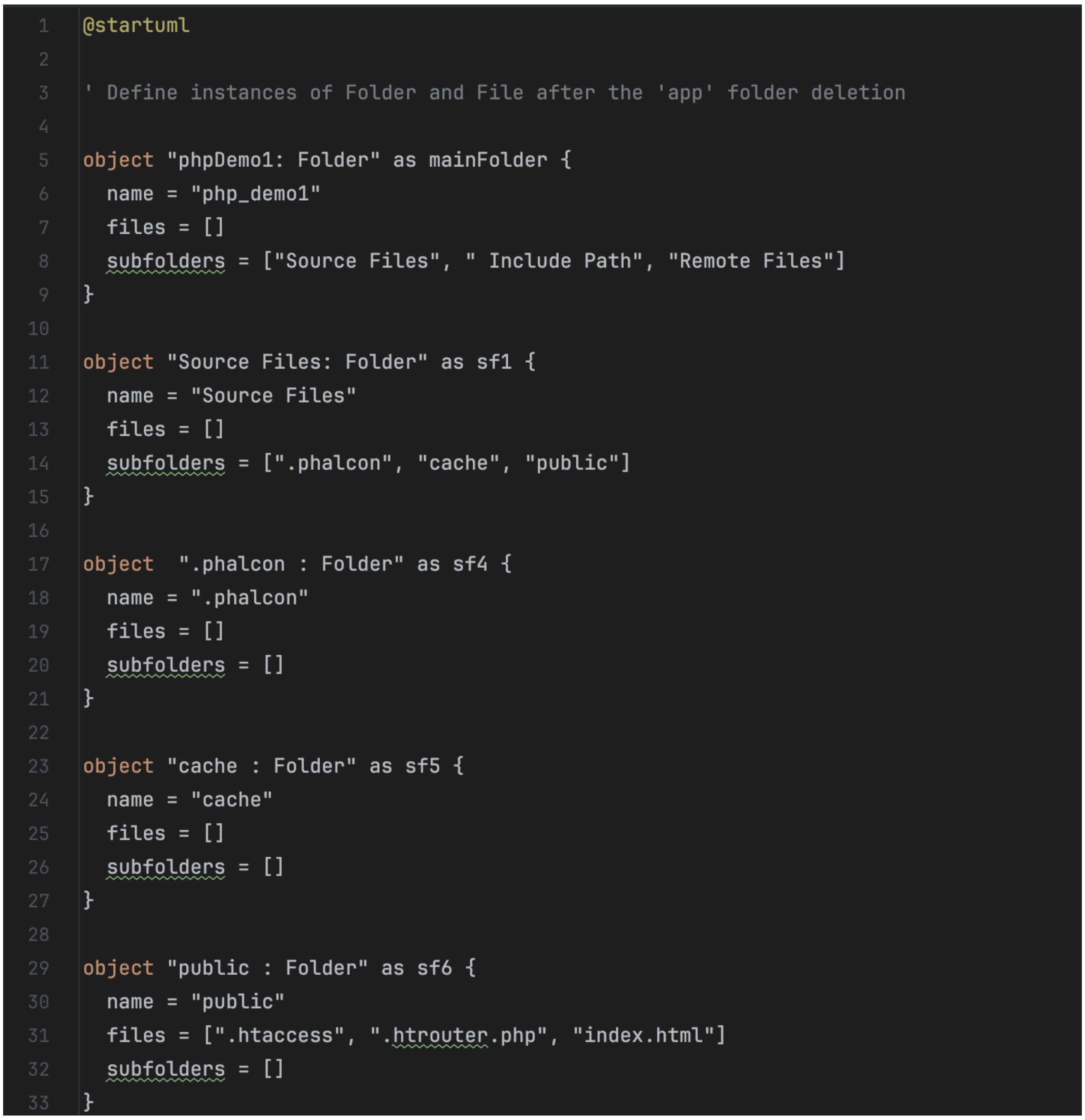


```
SourceFiles *-- Public

App *-- Config
App *-- Controllers
App *-- Library
App *-- Migrations
App *-- Models
App *-- Views

Public *-- htaccess
Public *-- htrouter
Public *-- indexHtml

@enduml
```

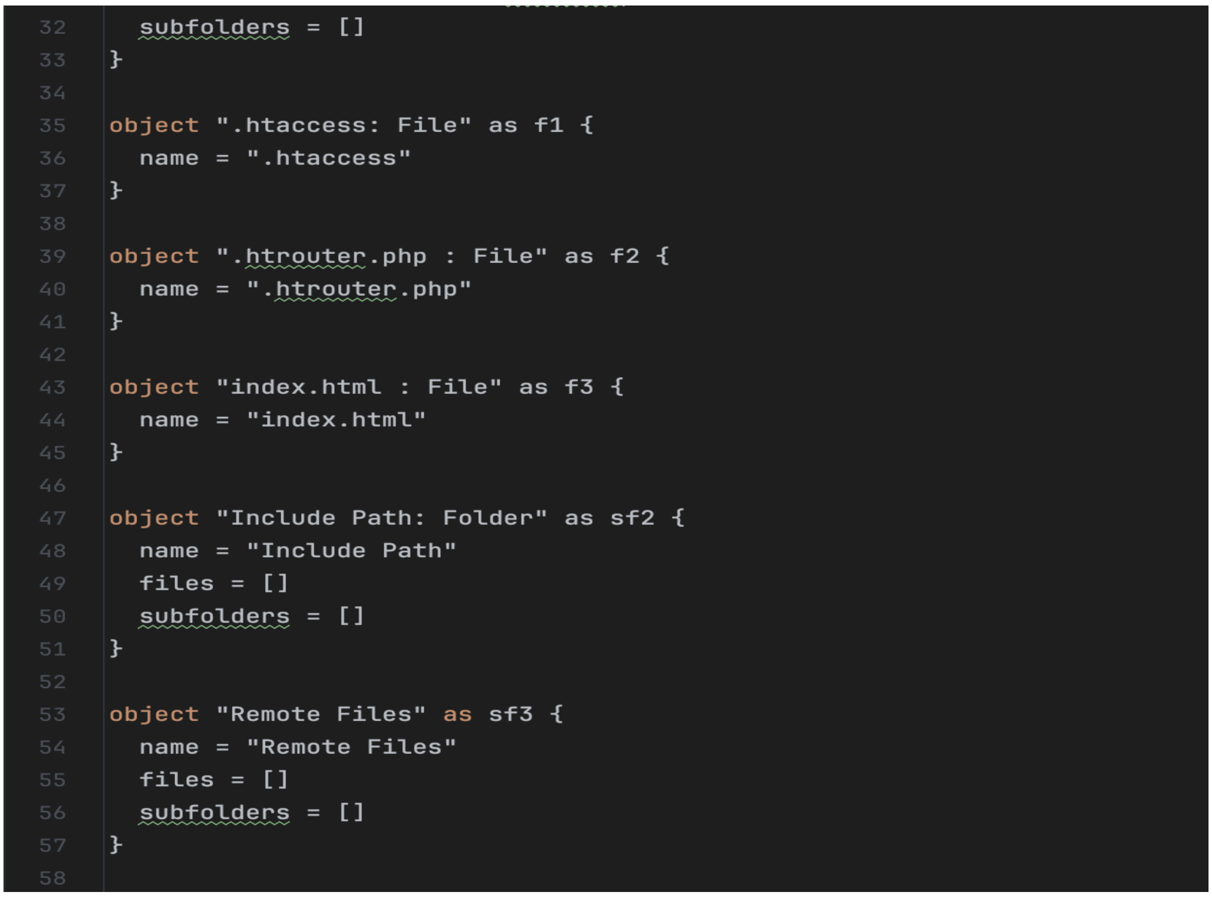
4a: Class Diagram - Composition


Folder
name: String
subFolders: Folders[*]
files: File[*]
getName(): String
setName(in name: String): void
removeSubfolder(in subFolderName: String): void
addSubfolder(in subFolderName: String): void
getSubfolder(in subFolderName: String): Folder
removeFile(in file: File): boolean
addFile(in fileName: String): void
printFolderContents(): void
toString(): String
File
name: String
getName(): String
setName(in name: String): void
toString(): String

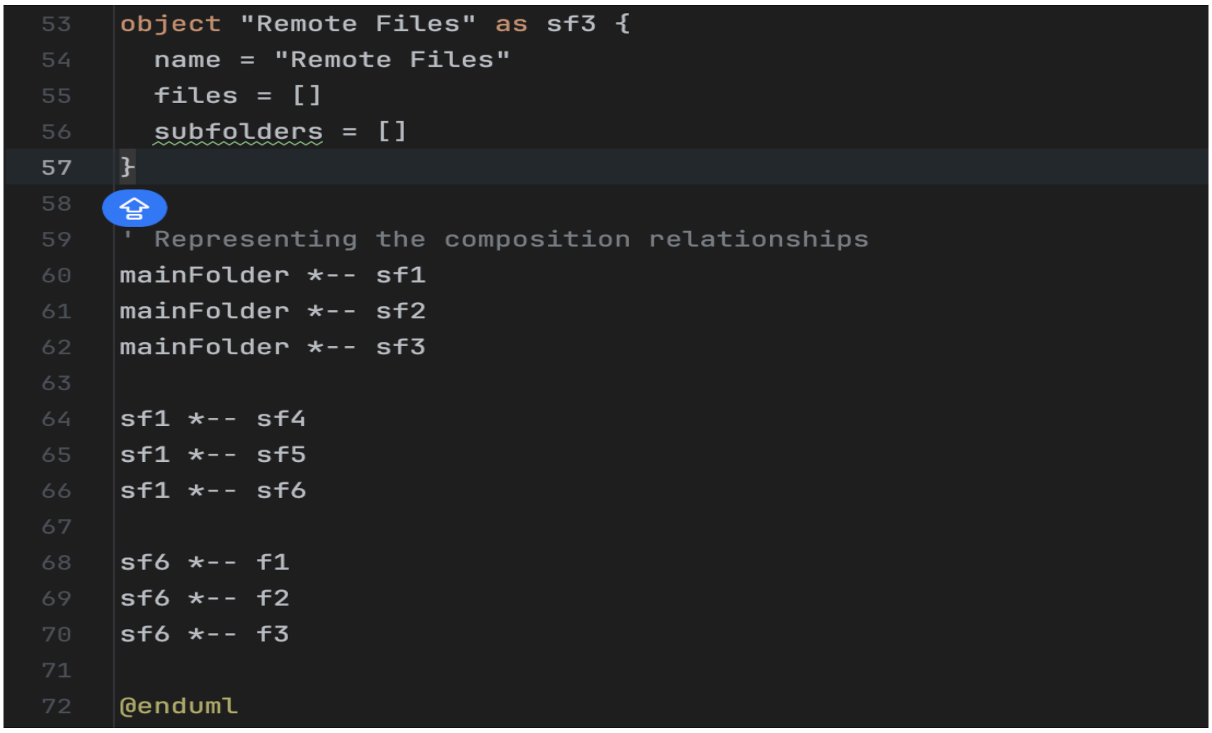
4b: Code for the Object Diagram



```
@startuml

' Define instances of Folder and File after the 'app' folder deletion

object "phpDemo1: Folder" as mainFolder {
  name = "php_demo1"
  files = []
  subfolders = ["Source Files", " Include Path", "Remote Files"]
}

object "Source Files: Folder" as sf1 {
  name = "Source Files"
  files = []
  subfolders = [".phalcon", "cache", "public"]
}

object  ".phalcon : Folder" as sf4 {
  name = ".phalcon"
  files = []
  subfolders = []
}

object "cache : Folder" as sf5 {
  name = "cache"
  files = []
  subfolders = []
}

object "public : Folder" as sf6 {
  name = "public"
  files = [".htaccess", ".htrouter.php", "index.html"]
  subfolders = []
}
```

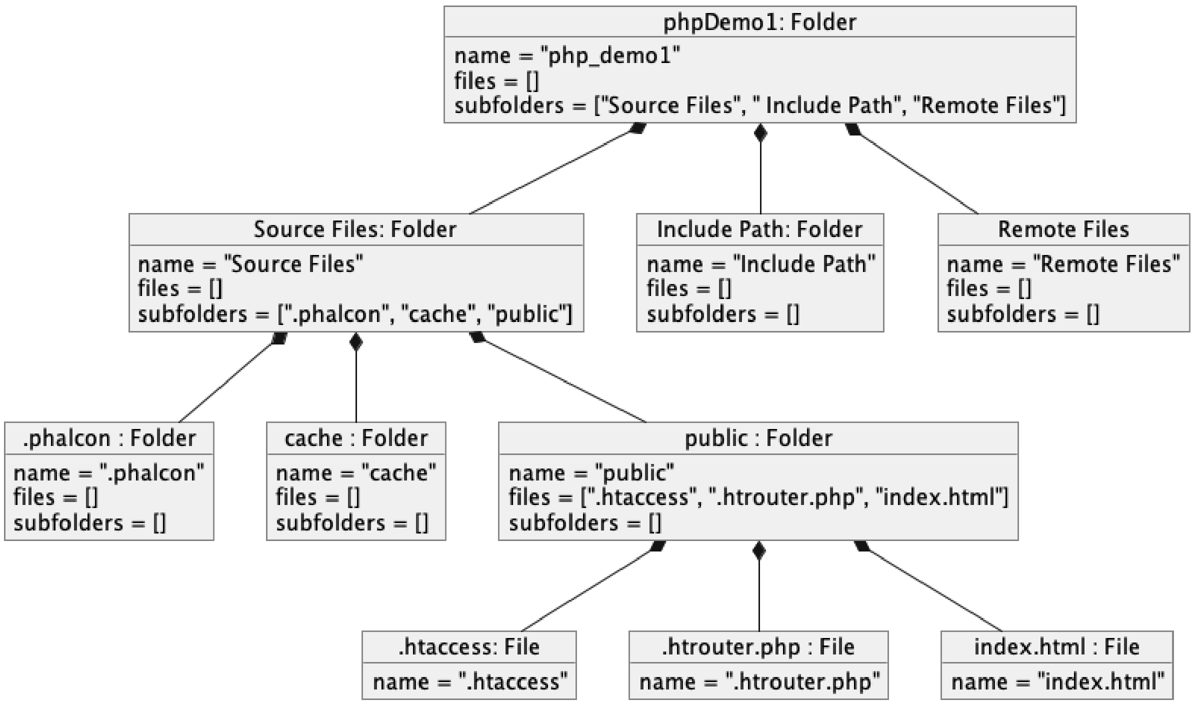


```
  subfolders = []
}

object ".htaccess: File" as f1 {
  name = ".htaccess"
}

object ".htrouter.php : File" as f2 {
  name = ".htrouter.php"
}

object "index.html : File" as f3 {
  name = "index.html"
}

object "Include Path: Folder" as sf2 {
  name = "Include Path"
  files = []
  subfolders = []
}

object "Remote Files" as sf3 {
  name = "Remote Files"
  files = []
  subfolders = []
}

```



```
object "Remote Files" as sf3 {
  name = "Remote Files"
  files = []
  subfolders = []
}

' Representing the composition relationships
mainFolder *-- sf1
mainFolder *-- sf2
mainFolder *-- sf3

sf1 *-- sf4
sf1 *-- sf5
sf1 *-- sf6

sf6 *-- f1
sf6 *-- f2
sf6 *-- f3

@enduml
```

4b: Object Diagram - Composition


phpDemo1: Folder
name = "php_demo1"
files = []
subfolders = ["Source Files", " Include Path", "Remote Files"]
Source Files: Folder
name = "Source Files"
files = []
subfolders = [".phalcon", "cache", "public"]
Include Path: Folder
name = "Include Path"
files = []
subfolders = []
Remote Files
name = "Remote Files"
files = []
subfolders = []
.phalcon : Folder
name = ".phalcon"
files = []
subfolders = []
cache : Folder
name = "cache"
files = []
subfolders = []
public : Folder
name = "public"
files = [".htaccess", ".htrouter.php", "index.html"]
subfolders = []
.htaccess: File
name = ".htaccess"
.htrouter.php : File
name = ".htrouter.php"
index.html : File
name = "index.html"