**PROJECT 4**

**Pseudocode:**

Plot class:

**Attributes:**

Integer x, y

Integer width, depth

**Constructors:**

**Plot()**

**Purpose:** Initialize the plot without any parameters. The upper left point is at (0, 0) and the width and length are 1.

**Plot(x: int, y: int, width: int, depth: int)**

**Purpose:** Initialize the plot with the given parameters. x and y give the upper left point coordinate, width gives the value for horizontal length, and depth gives the value for vertical length.

**Plot(other: Plot)**

**Purpose:** Copy constructor

**Methods:**

**getX() -> Integer**

Return x

**getY() -> Integer**

Return y

**getWidth() -> Integer**

Return width

**getDepth() -> Integer**

Return depth

**setX(newX: Integer) -> void**

x = newX

**setY(newY: Integer) -> void**

y = newY

**setWidth(newWidth: Integer) -> void**

width = newWidth

**setDepth(newDepth: Integer) -> void**

depth = newDepth

**overlaps(other: Plot) -> Boolean**

**Purpose:** Determine if this plot overlaps with the other plot.

If ((x + width <= other.x || other.x + other.width <= x

y + depth <= other.y || other.y + other.depth <= y)):

Return false (not overlapping)

Else

Return true (overlapping)

**encompasses(other: Plot) -> Boolean**

**Purpose:** Determine if this plot completely contains the other plot.

If this plot is outside of the other plot:

Return true

Else

Return false

**toString() -> String**

**Purpose:** Return a string in the format

Return the outputs as a formatted string

Property class:

**Attributes:**

String propertyName

String city

Double rentAmount

String owner

Plot plot

**Constructors:**

**Property()**

**Purpose:** Initialize the property without any parameters. All the string attributes are empty, and all the numerical attributes are 0

**Property(name: String, city: String, rent: double, owner: String)**

**Purpose:** Initialize the property without any parameters with the given parameters

**Property(name: String, city: String, rent: double, owner: String, x: integer, y: integer, width: integer, depth: integer)**

**Purpose:** Initialize the property without any parameters with the given parameters with the last four parameters initialized for plot class

**Property(other: Property)**

**Purpose:** Copy constructor

**Methods:**

**getPropertyName() -> String**

Return propertyName

**getCity() -> String**

Return city

**getRentAmount() -> Double**

Return rentAmount

**getOwner() -> String**

Return owner

**getPlot() -> Plot**

Return plot

**toString() -> String**

**Purpose:** Return a string representing the property.

Return the outputs as a formatted string

ManagementCompany class:

**Attributes:**

String name

String taxId

Double managementFee

Array[Property] properties

Plot plot

Integer numberOfProperties

**Constructors:**

**ManagementCompany()**

**Purpose:** Initialize the management company without any parameters. All the string attributes are empty, and all the numerical attributes are 0

**ManagementCompany(name: String, taxId: String, managementFee: double)**

**Purpose:** Initialize the management company with the given parameters

**ManagementCompany(name: String, taxId: String, managementFee: double, initialX: integer, initialY: integer, width: integer, depth: integer)**

**Purpose:** Initialize the management company with the given parameters, and the last 4 are used to initialize the plot constructors.

**ManagementCompany(other: ManagementCompany)**

**Purpose:** Copy constructor

**getName() -> String**

Return name

**getTaxId() -> String**

Return taxId

**getManagementFee() -> Double**

Return managementFee

**getPlot() -> Plot**

Return plot

**getProperties() -> Array[Property]**

Return properties array

**getPropertiesCount() -> Integer**

Return numberOfProperties

**addProperty(name: String, city: String, rent: double, owner): String -> Integer**

**Purpose:** Add a new property with the given parameters

Initialize a property object with the given parameters

Return the integer by calling the addProperty method with the newly created property object

**addProperty(name: String, city: String, rent: double, owner: String, x: int, y: int, width: int, depth: int) -> Integer**

**Purpose:** Add a new property with the given parameters, and the last 4 are used for initializing the plot constructor

Initialize a property object with the given parameters

Return the integer by calling the addProperty method with the newly created property object

**addProperty(other: Property) -> Integer**

**Purpose:** Adds a Property object to the array.

If the number of properties is bigger than Maximum number of properties (= 5)

Return -1

If the property object is null

Return -2

If the property object’s plot is not encompassed by the management company’s plot

Return -3

For a loop that loops the number of properties

If the property object overlaps any property elements in the array

Return -4

Add the property object to the array

Return number of properties

**getTotalRent() -> Double**

**Purpose:** Calculate the total rent of all properties.

For a loop that loops the number of properties

Sum up all property rent amounts

Return the total rent amount

**getHighestRentProperty() -> Property**

**Purpose:** Find the property with the highest rent.

For a loop that loops the number of properties

If the rent amount of the property element is bigger than the highest rent

Set the new highest rent amount

Return the property object with the highest rent

**removeLastProperty() -> void**

**Purpose:** Removes the last property.

If the number of properties is bigger than 0

Set the property element at subscript the current number of properties to null

Reduce the number of properties by 1

**isPropertiesFull() -> Boolean**

**Purpose:** Check if the properties array is full.

Return true if number of properties == maximum number of properties, otherwise false

**isManagementFeeValid() -> Boolean**

**Purpose:** Checks if management fee is within valid range (0-100%).

Return true if 0 <= management fee <= 100, otherwise false

**toString() -> String**

**Purpose:** Return information about all properties and total fee.

Return the outputs as a formatted string

**UML diagrams:**

|  |
| --- |
| Plot |
| -x: int  -y: int  -width: int  -depth: int |
| +Plot(): Plot  +Plot(x: int, y: int, width: int, depth: int): Plot  +Plot(otherPlot: Plot): Plot  +getX(): int  +getY(): int  +getWidth(): int  +getDepth(): int  +setX(x: int): int  +setY(y: int): int  +setWidth(width: int): int  +setDepth(depth: int): int  +overlaps(plot: Plot): boolean  +encompasses(plot: Plot): boolean  +toString(): String |

|  |
| --- |
| Property |
| -propertyName: String  -city: String  -owner: String  -rentAmount: double  -plot: Plot |
| +Property(): Property  +Property(name: String, city: String, rent: double, owner: String): Property  +Property(name: String, city: String, rent: double, owner: String, x: integer, y: integer, width: integer, depth: integer): Property  +Property(otherProperty: Property): Property  +getPropertyName(): String  +getCity(): String  +getOwner(): String  +getPlot(): Plot  +getRentAmount(): double  +toString(): String |

|  |
| --- |
| ManagementCompany |
| +MAX\_PROPERTY: int  +MGMT\_DEPTH: int  +MGMT\_WIDTH: int  -name: String  -taxID: String  -mgmtFeePercentage: double  -properties: Property[]  -plot: Plot  -numberOfProperties: int |
| +ManagementCompany(): ManagementCompany  +ManagementCompany(name: String, taxId: String, managementFee: double): ManagementCompany  +ManagementCompany(name: String, taxId: String, managementFee: double, initialX: integer, initialY: integer, width: integer, depth: integer): ManagementCompany  +ManagementCompany(otherCompany: ManagementCompany): ManagementCompany  +addProperty(name: String, city: String, rent: double, owner: String): int  +addProperty(name: String, city: String, rent: double, owner: String, x: int, y: int, width: int, depth: int): int  +addProperty(property: Property): int  +getName(): String  +getTaxID(): String  +getProperty(): Property[]  +getPropertiesCount(): int  +getMgmFeePer(): double  +getPlot(): Plot  +getTotalRent(): double  +getHighestRentProperty(): Property  +removeLastProperty(): void  +isPropertiesFull: boolean  +isManagementFeeValid(): boolean  +toString(): String |

**Test cases:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test cases # | Input | Expected output | Actual output | Did the test pass? |
| 1 | Real Estate  55555  5  House 5  Germantown  5555  Five Guys  0  0  5  5 | As shown in the screenshots | As shown in the screenshots | Yes |
| 2 | Apartment  Germantown  3000  John Smith  6  6  1  1 | As shown in the screenshots | As shown in the screenshots | Yes |
| 3 | Overlap  Germantown  4000  John Cena  3  3  3  3 | As shown in the screenshots | As shown in the screenshots | Yes |
| 4 | Encompass  Germantown  5000  Dwayne Johnson  8  8  3  3 | As shown in the screenshots | As shown in the screenshots | Yes |

**Screenshots:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Junit test cases:**

**A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer error message

Description automatically generatedA screenshot of a computer error

Description automatically generated**

**A screenshot of a computer error message

Description automatically generatedA screenshot of a computer error

Description automatically generated**

**GitHub Screenshots:**

**Lesson learned:**

Through this project, I have had the chance to deal with multiple classes and how to implement different types of constructors that require different arguments. Also, the logic after checking overlapping and encompassing is tricky and challenging without a visual aid. I had the chance to look into inserting pictures and images into Java in Eclipse as well.