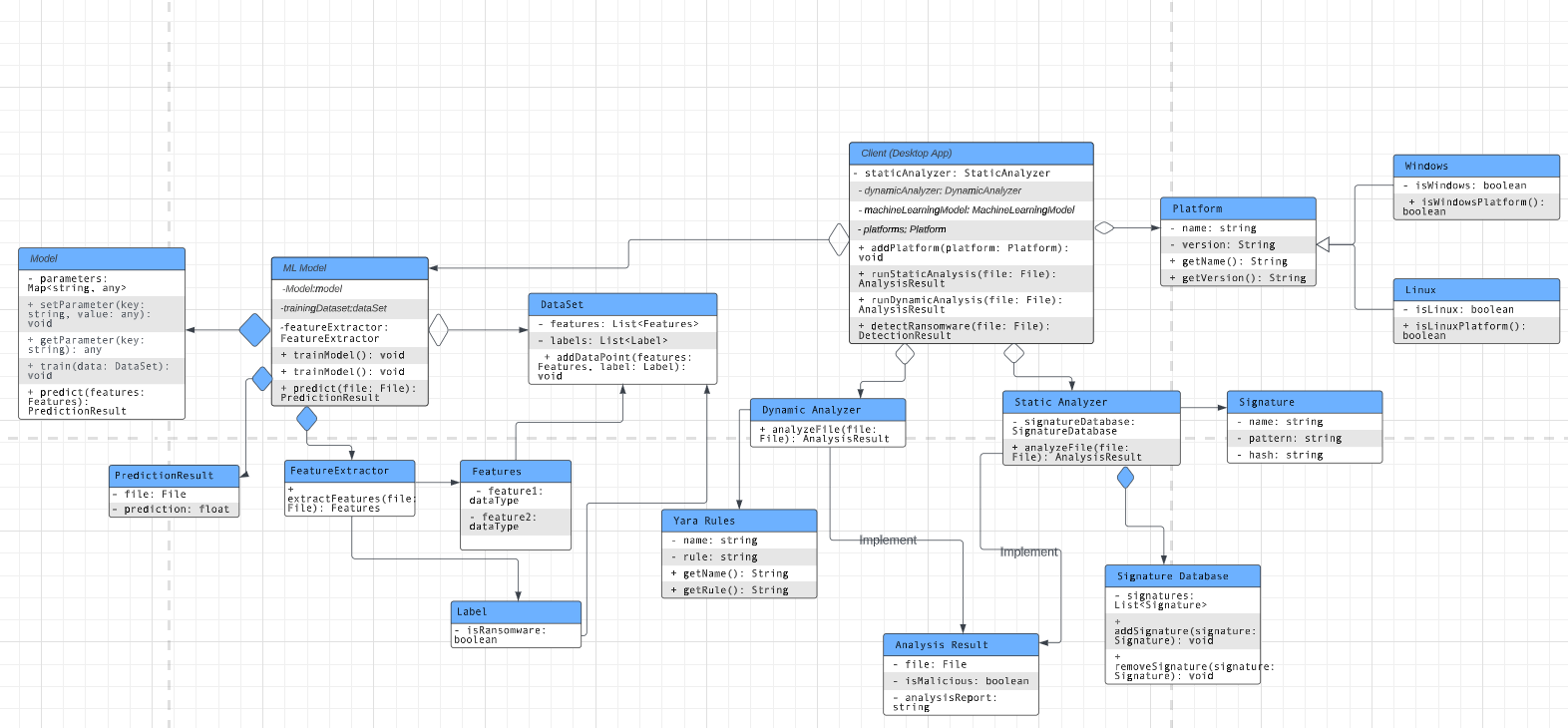
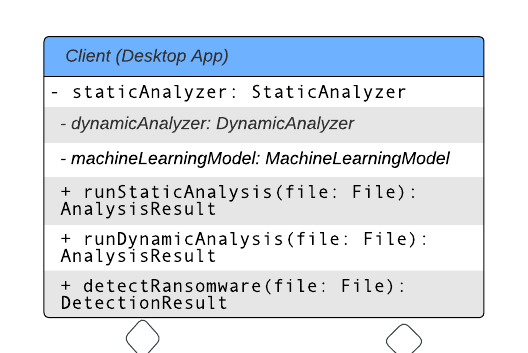
3.2 Class Diagram



3.3.1 Class: Client (Desktop App)

Fig:



Class: Client (Desktop App)

● Purpose: Represents the desktop application for the Ransomware Detection system.

● Constraints: None

● Persistent: No

3.3.1.1 Attribute Descriptions

1. Attribute: staticAnalyzer

Type: StaticAnalyzer

Description: An instance of the StaticAnalyzer class used for static analysis of files.

Constraints: None.

2. Attribute: dynamicAnalyzer

Type: DynamicAnalyzer

Description: An instance of the DynamicAnalyzer class used for dynamic analysis of files.

Constraints: None.

3. Attribute: machineLearningModel

Type: MachineLearningModel

Description: An instance of the MachineLearningModel class used for predicting whether a file is ransomware or not.

Constraints: None.

3.3.1.2 Method Descriptions

1. Method: runStaticAnalysis(file: File)

Return Type: AnalysisResult

Parameters: file (File)

Return value: The result of the static analysis as an AnalysisResult object.

Pre-condition: The StaticAnalyzer instance must be initialized.

Post-condition: Returns the analysis result of the file.

Attributes read/used: staticAnalyzer

Methods called: StaticAnalyzer.analyzeFile(file: File)

Processing logic: Calls the analyzeFile method of the StaticAnalyzer instance to perform static analysis on the file and returns the analysis result.

2. Method: runDynamicAnalysis(file: File)

Return Type: AnalysisResult

Parameters: file (File)

Return value: The result of the dynamic analysis as an AnalysisResult object.

Pre-condition: The DynamicAnalyzer instance must be initialized.

Post-condition: Returns the analysis result of the file.

Attributes read/used: dynamicAnalyzer

Methods called: DynamicAnalyzer.analyzeFile(file: File)

Processing logic: Calls the analyzeFile method of the DynamicAnalyzer instance to perform dynamic analysis on the file and returns the analysis result.

3. Method: detectRansomware(file: File)

Return Type: DetectionResult

Parameters: file (File)

Return value: The result of the ransomware detection as a DetectionResult object.

Pre-condition: The MachineLearningModel instance must be initialized.

Post-condition: Returns the detection result of the file.

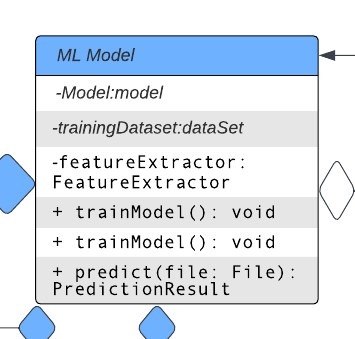
Attributes read/used: machineLearningModel

Methods called: MachineLearningModel.predict(features: Features)

Processing logic: Extracts features from the file, creates a Features object, and passes it to the predict method of the MachineLearningModel instance to predict whether the file is ransomware or not. Returns the detection result.

3.3.1 Class: MachineLearningModel

Fig.



Class: MachineLearningModel

● Purpose: Represents the machine learning model used for ransomware detection.

● Constraints: None

● Persistent: No

3.3.1.1 Attribute Descriptions

1. Attribute: model

Type: Model

Description: An instance of the Model class representing the machine learning model implementation.

Constraints: None.

2. Attribute: trainingData

Type: DataSet

Description: An instance of the DataSet class representing the training data used to train the machine learning model.

Constraints: None.

3. Attribute: featureExtractor

Type: FeatureExtractor

Description: An instance of the FeatureExtractor class used for extracting features from files.

Constraints: None.

3.3.1.2 Method Descriptions

1. Method: trainModel()

Return Type: void

Parameters: None

Return value: None

Pre-condition: The model and trainingData attributes must be initialized.

Post-condition: Trains the machine learning model using the training data.

Attributes read/used: model, trainingData

Methods called: Model.train(data: DataSet)

Processing logic: Calls the train method of the Model instance, passing the training data, to train the machine learning model.

2. Method: predict(file: File)

Return Type: PredictionResult

Parameters: file (File)

Return value: The prediction result as a PredictionResult object.

Pre-condition: The model, trainingData, and featureExtractor attributes must be initialized.

Post-condition: Returns the prediction result of the file.

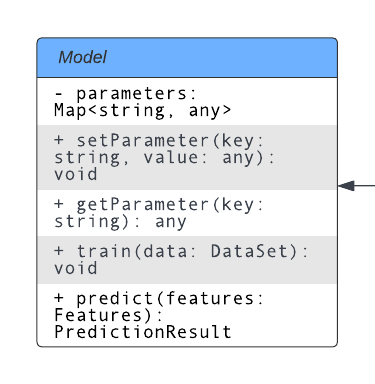
Attributes read/used: model, featureExtractor

Methods called: FeatureExtractor.extractFeatures(file: File), Model.predict(features: Features)

Processing logic: Uses the featureExtractor to extract features from the file, creates a Features object, and passes it to the predict method of the Model instance to predict whether the file is ransomware or not. Returns the prediction result as a PredictionResult object.

3.3.2 Class: Model

Fig.



Class: Model

● Purpose: Represents a machine learning model used for ransomware detection.

● Constraints: None

● Persistent: No

3.3.2.1 Attribute Descriptions

1. Attribute: parameters

Type: Map<string, any>

Description: A map that holds the parameters of the model.

Constraints: None.

3.3.2.2 Method Descriptions

1. Method: setParameter(key: string, value: any)

Return Type: void

Parameters: key (string), value (any)

Return value: None

Pre-condition: None

Post-condition: Sets the parameter with the given key to the specified value.

Attributes read/used: parameters

Methods called: None

Processing logic: Sets the value of the parameter in the parameters map with the provided key.

2. Method: getParameter(key: string)

Return Type: any

Parameters: key (string)

Return value: The value of the parameter associated with the given key.

Pre-condition: None

Post-condition: Returns the value of the parameter if it exists, otherwise returns null.

Attributes read/used: parameters

Methods called: None

Processing logic: Retrieves the value of the parameter from the parameters map based on the provided key.

3. Method: train(data: DataSet)

Return Type: void

Parameters: data (DataSet)

Return value: None

Pre-condition: None

Post-condition: Trains the model using the provided training data.

Attributes read/used: None

Methods called: None

Processing logic: Trains the machine learning model using the data from the given DataSet object.

4. Method: predict(features: Features)

Return Type: PredictionResult

Parameters: features (Features)

Return value: The prediction result as a PredictionResult object.

Pre-condition: None

Post-condition: Returns the prediction result based on the given features.

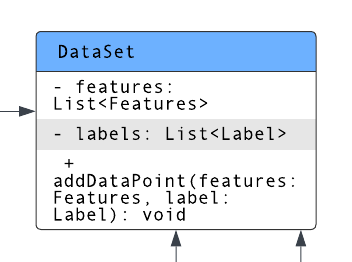
Attributes read/used: None

Methods called: None

Processing logic: Uses the machine learning model to predict the result based on the provided features and returns the prediction result.

3.3.1 Class: DataSet

Fig.



Class: DataSet

● Purpose: Stores the training data used for machine learning model training.

● Constraints: None

● Persistent: No

3.3.1.1 Attribute Descriptions

1. Attribute: features

Type: List<Features>

Description: A list of Features objects representing the extracted features from files.

Constraints: None.

2. Attribute: labels

Type: List<Label>

Description: A list of Label objects representing the corresponding labels for the features.

Constraints: None.

3.3.1.2 Method Descriptions

1. Method: addDataPoint(features: Features, label: Label)

Return Type: void

Parameters: features (Features), label (Label)

Return value: None

Pre-condition: None

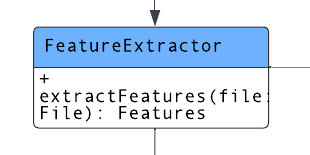
Post-condition: Adds a data point (features and label) to the dataset.

Attributes read/used: features, labels

Methods called: None

Processing logic: Adds the provided features and label as a new data point to the dataset by appending them to the respective lists.

3.3.2 Class: FeatureExtractor

Fig. 

Class: FeatureExtractor

● Purpose: Extracts features from files for ransomware detection.

● Constraints: None

● Persistent: No

3.3.2.1 Attribute Descriptions

1. Attribute: -

Type: -

Description: -

Constraints: -

3.3.2.2 Method Descriptions

1. Method: extractFeatures(file: File)

Return Type: Features

Parameters: file (File)

Return value: The extracted features as a Features object.

Pre-condition: None

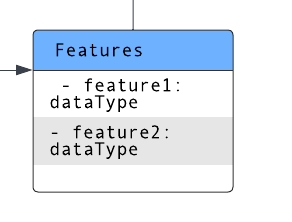
Post-condition: Returns the extracted features from the file.

Attributes read/used: -

Methods called: -

Processing logic: Takes a file as input and performs the necessary operations to extract features from the file. Creates a Features object containing the extracted features and returns it.

3.3.1 Class: Features

Fig. 

Class: Features

● Purpose: Represents the features extracted from a file.

● Constraints: None

● Persistent: No

3.3.1.1 Attribute Descriptions

1. Attribute: feature1

Type: dataType

Description: Represents the value of feature1 for the file.

Constraints: None.

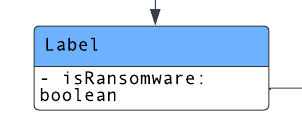
2. Attribute: feature2

Type: dataType

Description: Represents the value of feature2 for the file.

Constraints: None.

3.3.7 Class: Label

Fig. 

Class: Label

● Purpose: Represents the label associated with a data point indicating whether it is classified as ransomware or not.

● Constraints: None

● Persistent: No

3.3.7.1 Attribute Descriptions

1. Attribute: isRansomware

Type: boolean

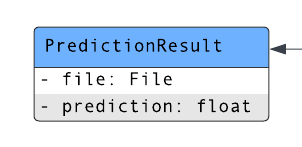
Description: Indicates whether the data point is classified as ransomware (true) or not (false).

Constraints: None.

3.3.7.2 Method Descriptions

No methods are defined for the Label class.

3.3.2 Class: PredictionResult

Fig. Class: PredictionResult

● Purpose: Represents the result of a prediction made by the machine learning model.

● Constraints: None

● Persistent: No

3.3.2.1 Attribute Descriptions

1. Attribute: file

Type: File

Description: The file for which the prediction was made.

Constraints: None.

2. Attribute: prediction

Type: float

Description: The prediction value (e.g., probability) generated by the model.

Constraints: None.

3.3.2.2 Method Descriptions

1. Method: PredictionResult(file: File, prediction: float)

Return Type: PredictionResult

Parameters:

- file (File): The file for which the prediction was made.

- prediction (float): The prediction value generated by the model.

Return value: Creates a PredictionResult instance with the given file and prediction.

Pre-condition: -

Post-condition: Returns a new PredictionResult object with the attributes set.

Attributes read/used: -

Methods called: -

Processing logic: This constructor initializes the attributes of the PredictionResult instance with the provided file and prediction values.

2. Method: getFile()

Return Type: File

Parameters: -

Return value: The file for which the prediction was made.

Pre-condition: -

Post-condition: Returns the file attribute.

Attributes read/used: file

Methods called: -

Processing logic: Returns the value of the file attribute.

3. Method: getPrediction()

Return Type: float

Parameters: -

Return value: The prediction value generated by the model.

Pre-condition: -

Post-condition: Returns the prediction attribute.

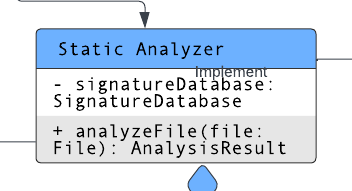
Attributes read/used: prediction

Methods called: -

Processing logic: Returns the value of the prediction attribute.

3.3.2 Class: StaticAnalyzer

Fig.



Class: StaticAnalyzer

● Purpose: Performs static analysis of files using a signature database.

● Constraints: None

● Persistent: No

3.3.2.1 Attribute Descriptions

1. Attribute: signatureDatabase

Type: SignatureDatabase

Description: An instance of the SignatureDatabase class that holds a collection of signatures for static analysis.

Constraints: None.

3.3.2.2 Method Descriptions

1. Method: analyzeFile(file: File)

Return Type: AnalysisResult

Parameters: file (File)

Return value: The result of the analysis as an AnalysisResult object.

Pre-condition: The SignatureDatabase instance must be initialized.

Post-condition: Returns the analysis result of the file.

Attributes read/used: signatureDatabase

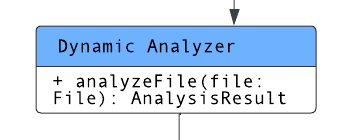
Methods called: SignatureDatabase.addSignature(signature: Signature), SignatureDatabase.removeSignature(signature: Signature)

Processing logic: Analyzes the given file by checking its content against the signatures in the SignatureDatabase. If a match is found, the file is considered malicious. Returns the analysis result including the file, a boolean indicating whether the file is considered malicious, and an analysis report.

3.3.2.3 Class Collaborations

1. SignatureDatabase: The StaticAnalyzer class collaborates with the SignatureDatabase class by using its methods to add and remove signatures for static analysis.

3.3.2 Class: DynamicAnalyzer

Fig. 

Class: DynamicAnalyzer

● Purpose: Performs dynamic analysis of files during runtime.

● Constraints: None

● Persistent: No

3.3.2.1 Attribute Descriptions

1. Attribute: None

3.3.2.2 Method Descriptions

1. Method: analyzeFile(file: File)

Return Type: AnalysisResult

Parameters: file (File)

Return value: The result of the dynamic analysis as an AnalysisResult object.

Pre-condition: None

Post-condition: Returns the analysis result of the file.

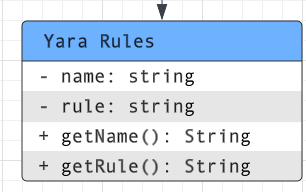
Attributes read/used: None

Methods called: None

Processing logic: This method performs dynamic analysis on the given file during runtime. It may involve executing the file in a controlled environment and monitoring its behavior. The analysis result, including whether the file is considered malicious and any relevant analysis report, is returned as an AnalysisResult object.

3.3.2 Class: Yara Rules

Fig.



Class: Yara Rules

● Purpose: Represents a set of rules for detecting malware using the YARA tool.

● Constraints: None

● Persistent: No

3.3.1.1 Attribute Descriptions

1. Attribute: name

Type: string

Description: The name of the rule set.

Constraints: None.

1. Attribute: rule

Type: string

Description: The rule expression in YARA syntax.

Constraints: Must be a valid YARA rule.

3.3.1.2 Method Descriptions

1. Method: getName()

Return Type: string

Parameters: None

Return value: The name of the rule set.

Pre-condition: None

Post-condition: Returns the name attribute.

Attributes read/used: name

Methods called: None

Processing logic: Returns the name attribute.

1. Method: getRule()

Return Type: string

Parameters: None

Return value: The rule expression in YARA syntax.

Pre-condition: None

Post-condition: Returns the rule attribute.

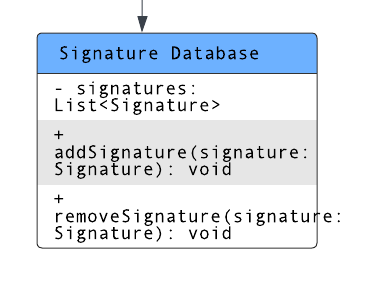
Attributes read/used: rule

Methods called: None

Processing logic: Returns the rule attribute.

3.3.2 Class: SignatureDatabase

Fig.



Class: SignatureDatabase

● Purpose: Manages the collection of signatures used by the static analyzer.

● Constraints: None

● Persistent: No

3.3.2.1 Attribute Descriptions

1. Attribute: signatures[]

Type: Array of Signature

Description: Stores the list of signatures in the signature database.

Constraints: None.

3.3.2.2 Method Descriptions

1. Method: addSignature(signature: Signature)

Return Type: void

Parameters: signature (Signature)

Return value: None

Pre-condition: None

Post-condition: Adds the provided signature to the signature database.

Attributes read/used: signatures

Methods called: None

Processing logic: Adds the provided signature to the list of signatures in the signature database.

2. Method: removeSignature(signature: Signature)

Return Type: void

Parameters: signature (Signature)

Return value: None

Pre-condition: None

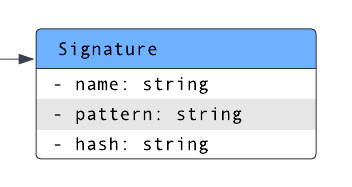
Post-condition: Removes the provided signature from the signature database.

Attributes read/used: signatures

Methods called: None

Processing logic: Removes the provided signature from the list of signatures in the signature database.

3.3.2 Class: Signature

Fig. 

Class: Signature

● Purpose: Represents a signature used by the StaticAnalyzer to identify specific patterns in files.

● Constraints: None

● Persistent: No

3.3.2.1 Attribute Descriptions

1. Attribute: name

Type: string

Description: The name or identifier of the signature.

Constraints: None.

2. Attribute: pattern

Type: string

Description: The specific pattern used for matching in the file.

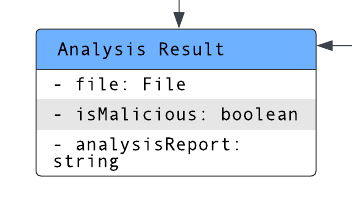
Constraints: None.

3.3.2.2 Method Descriptions

No methods are specified for the Signature class in the given context.

3.3.2 Class: AnalysisResult

Fig



Class: AnalysisResult

● Purpose: Represents the result of an analysis performed on a file.

● Constraints: None.

● Persistent: No.

3.3.2.1 Attribute Descriptions

1. Attribute: file

Type: File

Description: The file that was analyzed.

Constraints: None.

2. Attribute: isMalicious

Type: boolean

Description: Indicates whether the file is considered malicious or not.

Constraints: None.

3. Attribute: analysisReport

Type: string

Description: A report providing details of the analysis performed on the file.

Constraints: None.

3.3.2.2 Method Descriptions

1. Method: AnalysisResult()

Return Type: AnalysisResult

Parameters: None

Return value: Creates an AnalysisResult instance.

Pre-condition: Called when creating an analysis result for a file.

Post-condition: Initializes the attributes of the AnalysisResult instance.

Attributes read/used: None.

Methods called: None.

Processing logic: This is the constructor of the AnalysisResult class, called when creating an instance of the class. It sets the initial values of the attributes.

2. Method: getFile()

Return Type: File

Parameters: None

Return value: The file that was analyzed.

Pre-condition: The AnalysisResult instance must be initialized.

Post-condition: Returns the file attribute.

Attributes read/used: file

Methods called: None.

Processing logic: Returns the file attribute of the AnalysisResult instance.

3. Method: isMaliciousFile()

Return Type: boolean

Parameters: None

Return value: Indicates whether the file is considered malicious or not.

Pre-condition: The AnalysisResult instance must be initialized.

Post-condition: Returns the isMalicious attribute.

Attributes read/used: isMalicious

Methods called: None.

Processing logic: Returns the isMalicious attribute of the AnalysisResult instance.

4. Method: getAnalysisReport()

Return Type: string

Parameters: None

Return value: The analysis report providing details of the analysis performed on the file.

Pre-condition: The AnalysisResult instance must be initialized.

Post-condition: Returns the analysisReport attribute.

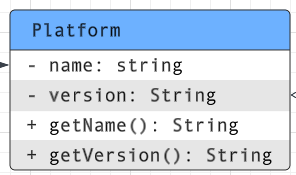
Attributes read/used: analysisReport

Methods called: None.

Processing logic: Returns the analysisReport attribute of the AnalysisResult instance.

3.3.2 Class: Platform

Fig.



Class: Platform

● Purpose: Represents a file in the file system.

● Constraints: None

● Persistent: Yes

3.3.1.1 Attribute Descriptions

1. Attribute: name

Type: string

Description: The name of the file.

Constraints: None.

1. Attribute: size

Type: string

Description: The size of the file in bytes.

Constraints: None.

1. Attribute: type

Type: string

Description: The type of the file, such as .txt, .pdf, .exe, etc.

Constraints: None.

3.3.1.2 Method Descriptions

1. Method: getName()

Return Type: string

Parameters: None

Return value: The name of the file.

Pre-condition: None

Post-condition: Returns the name attribute.

Attributes read/used: name

Methods called: None

Processing logic: Returns the name attribute.

1. Method: getSize()

Return Type: string

Parameters: None

Return value: The size of the file in bytes.

Pre-condition: None

Post-condition: Returns the size attribute.

Attributes read/used: size

Methods called: None

Processing logic: Returns the size attribute.

1. Method: getType()

Return Type: string

Parameters: None

Return value: The type of the file, such as .txt, .pdf, .exe, etc.

Pre-condition: None

Post-condition: Returns the type attribute.

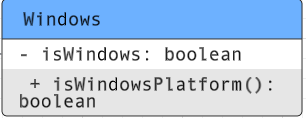
Attributes read/used: type

Methods called: None

Processing logic: Returns the type attribute.

3.3.2 Class: Windows

Fig.



Class: Windows

● Purpose: To check if the operating system is Windows using the System.getProperty() method.

● Constraints: None

● Persistent: No

3.3.1.1 Attribute Descriptions

1. Attribute: isWindows

Type: boolean

Description: A boolean variable that stores the result of the check.

Constraints: None.

3.3.1.2 Method Descriptions

1. Method: isWindowsPlatform()

Return Type: boolean

Parameters: None

Return value: The value of the isWindows attribute.

Pre-condition: None

Post-condition: Returns the value of the isWindows attribute.

Attributes read/used: isWindows

Methods called: System.getProperty(String key)

Processing logic: Assigns the value of the isWindows attribute to the result of comparing the system property “os.name” with “Windows”. Returns the value of the isWindows attribute.