

Module 5: Feature Objects

This module introduces you to feature objects and methods. You will learn how to transverse features, collect feature information, delete, and determine feature type in multiple models. You will utilize object orient programming techniques for feature manipulation.

Objectives

After completing this module you will be able to:

- Understand the feature objects
- Retrieve feature information
- Determine feature types
- Delete features
- Rename features
- Manipulate feature geometry





Features Objects

Visiting Features in Models

Objectives

- To understand Feature Objects
- To understand how to develop feature classes
- To understand how to traverse through models
- To understand how to select features type
- To understand how to retrieve feature information
- To understand how to remove features
- To understand how to rename features
- To understand how to place features

Outline

- 8.1 Introduction
- 8.2 Feature Objects
- 8.3 Obtaining Feature Information

8.1 Introduction

Many companies would like to manipulate feature placements and to retrieve information about certain features from company sales configurators. Sometimes, users would like to extract several hole locations from a component to a formatted file for manufacturability. Some companies have strict feature naming standards; therefore legacy model data features must be renamed. A J-Link application can fulfill these tasks.

8.2 Feature Objects

All Pro/ENGINEERS models are made up of features. The methods in the below table show the model feature access methods.

Feature Methods – Accessing Feature Objects	
Methods	Description
pfcFeature.Feature.ListChildren	Return a sequence of features that contains all the children
pfcFeature.Feature.ListParents	Return a sequence of features that contain all the parents
pfcFeature.FeatureGroup.GetGroupLeader	Get the first feature in a group
pfcFeature.FeatureGroup.GetPatternLeader	Get the first feature of a pattern
pfcFeature.FeaturePattern.ListMembers	Return features of a pattern
pfcSolid.Solid.ListFailedFeatures	Returns a sequence that contains all features that failed regeneration
pfcSolid.Solid.ListFeaturesByType	Returns a sequence of features contained in the model. You can specify which type of feature to collect by passing in one of the FeatureType enumeration objects, or you can collect all features by passing null as the type. If you list all features, the resulting sequence will include invisible features that Pro/ENGINEER creates internally. Use the method's <i>VisibleOnly</i> argument to exclude them.
pfcSolid.Solid.GetFeatureById	Return the feature object with the corresponding integer identifier

Feature Information

Feature Methods – Feature Objects	
Methods	Description
pfFeature.Feature.GetFeatType	Returns the type of a feature
pfFeature.Feature.GetStatus	Returns whether the feature is suppressed, active, or failed regeneration
pfFeature.Feature.GetIsVisible	Identifies whether the feature will be visible on the screen
pfFeature.Feature.GetIsReadOnly	Identifies whether the feature can be modified
pfFeature.Feature.GetNumber	Returns the feature regeneration number. This method returns null if the feature is suppressed
pfFeature.Feature.GetFeatTypeName	Returns a string representation of the feature type
pfFeature.Feature.GetFeatSubType	Returns a string representation of the feature sub-type, example: "Extrude" for protrusion features.

There are more feature objects that are explained and listed in the J-Link user guide. A developer can manipulate UDF features as well. Please refer to the J-Link user guide for more information about UDF objects if interested.

8.3 Obtaining Feature Information

Many users would like to traverse through a model to retrieve and extract information of specified features. The next exercise will collect feature information of randomly placed holes in a model. The task will instruct you to retrieve the hole features information and dimensions.

Task 8.3.1: Creating the FeatureOptions Class and method

1. Create a new class called FeatureOptions
2. Import required packages
3. Add a method named getHoleFeaturesDims
4. Add the try-catch block
5. Add code to retrieve Session
6. Add code to retrieve holes features
7. Add code to loop the hole features
8. Add code to get hole feature handle
9. Add code to retrieve feature model items
10. Add code to retrieve feature name
11. Add code to retrieve parents of features
12. Add code to retrieve each feature dimension
13. Compile the class by saving and update the JAR file

FeatureOptions .java – Creating FeatureOptions class and method

```
package com.felco.exampleapp;
```

```
import com.ptc.cipjava.*;
import com.ptc.pfc.pfcSession.Session;
import com.ptc.pfc.pfcSolid.*;
import com.ptc.pfc.pfcFeature.*;
import com.ptc.pfc.pfcModellItem.*;
import com.ptc.pfc.pfcDimension.*;
```

2. Import required packages

```
public class FeatureOptions {
```

3. Add Method

```
public void getHoleFeaturesDims(Solid solid) {
```

```
try {
```

4. Add try/catch block

```
// Get Session object from the solid
```

```
Session curSession = (Session) solid.GetDBParent();
```

5. Get Session

```
// List all holes in the solid model
```

```
Features hole_features = solid.ListFeaturesByType(Boolean.TRUE, FeatureType.FEATTYPE_HOLE);
```

6. Add code to retrieve holes

```
for (int i = 0; i < hole_features.getarraysize(); i++) {
```

7. Add code to loop through each hole feature

```
// Get hole features
```

```
Feature hole_feat = hole_features.get(i);
```

8. Get feature

```
// list all dimensions in the feature
```

```
ModellItems dimensions = hole_feat.ListSubItems(ModellItemType.ITEM_DIMENSION);
```

9. Add code to get feature model items

```
// Get feature type
```

```
String holeName = hole_feat.GetFeatTypeName() + (i + 1);
```

10. Get feature name

```
// Shown hole name in a dialog
```

```
curSession.UIShowMessageDialog(holeName, null);
```

```
// Get Features parents
```

```
Features fparents = hole_feat.ListParents();
```

```
for (int k = 0; k < fparents.getarraysize(); k++) {
```

```
Feature fFeature = fparents.get(k);
```

```
String sFeatureName = fFeature.GetName();
```

```
String sFeatureType = fFeature.GetFeatTypeName();
```

```
curSession.UIShowMessageDialog(sFeatureType + " : " + sFeatureName, null);
```

```
} // End of for
```

11. Add code to retrieve parents of feature

```

// Get dimensions values and symbols
for (int j = 0; j < dimensions.getarraysize(); j++) {
    // Get dimension of feature
    Dimension dim = (Dimension) dimensions.get(j);
    // Get dimension value
    double dimValue = dim.GetDimValue();
    // Get dimension symbol name
    String dimSymbol = dim.GetSymbol();

    // determine if the dimension is a Linear dim.
    if (dim.GetDimType().equals(DimensionType.DIM_LINEAR)) {
        curSession.UIShowMessageDialog(dimSymbol + " : " + dimValue, null);
    } // End of if
    // determine if the dimension is a diameter dim.
    if (dim.GetDimType().equals(DimensionType.DIM_DIAMETER)) {
        curSession.UIShowMessageDialog("Diameter" + " : " + dimValue, null);
    } // End of if
} // End of for
} // End of for

} // End of try
catch (jxthrowable x) {
}
} // End of method

} // End of class

```

12. Add code to retrieve feature dimension

Calling the FeatureOption method

Task 8.3.2: Calling the FeatureOptions class

1. Open and edit the MainApp.java
2. Add code for the FeatureOptions instance
3. Add conditional statement to determine model type to be a part.
4. Add code for calling the getHoleFeatureDims method
5. Compile the class by saving and update the JAR file
6. Open Pro/E and select JLink App to open the **plate.prt** file
7. Review the dialog boxes that appear