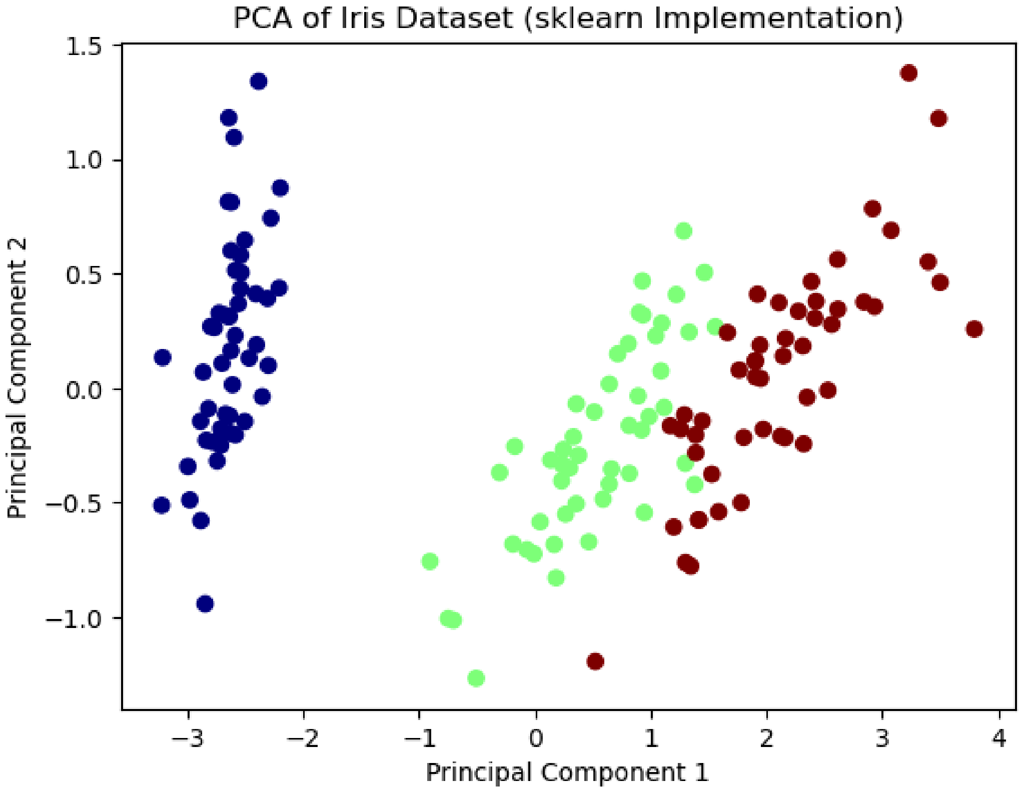
11a. PCA - Principal Component Analysis

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA as SklearnPCA

# Load the Iris dataset
X = load_iris().data
y = load_iris().target

# Perform PCA using sklearn
pca = SklearnPCA(n_components=2)
X_projected = pca.fit_transform(X)

print("Shape of Data:", X.shape)
print("Shape of transformed Data:", X_projected.shape)

# Plot the results
pc1 = X_projected[:, 0]
pc2 = X_projected[:, 1]

plt.scatter(pc1, pc2, c=y, cmap="jet")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.title("PCA of Iris Dataset (sklearn Implementation)")
plt.show()
```

```
Shape of Data: (150, 4)
Shape of transformed Data: (150, 2)
```

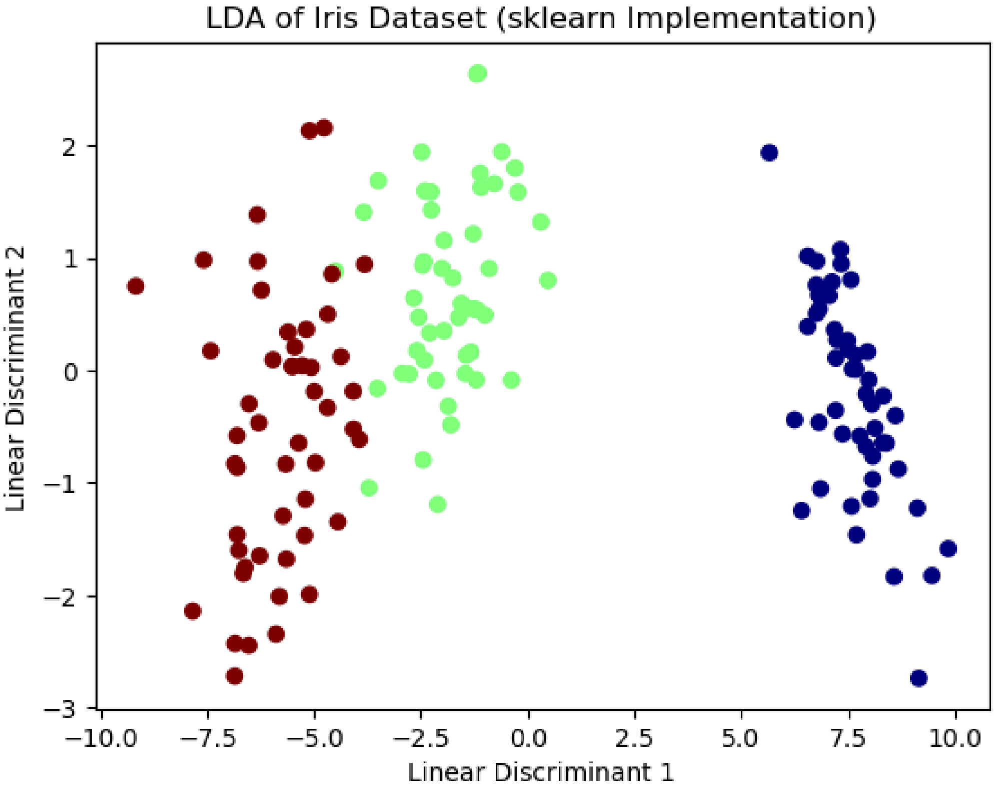


11b. LDA - Linear Discriminant Analysis

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# Load the Iris dataset
X = load_iris().data
y = load_iris().target

# Perform LDA using sklearn
lda = LinearDiscriminantAnalysis(n_components=2)
X_projected = lda.fit_transform(X, y)

print("Shape of Data:", X.shape)
print("Shape of transformed Data:", X_projected.shape)

# Plot the results
ld1 = X_projected[:, 0]
ld2 = X_projected[:, 1]

plt.scatter(ld1, ld2, c=y, cmap="jet")
plt.xlabel("Linear Discriminant 1")
plt.ylabel("Linear Discriminant 2")
plt.title("LDA of Iris Dataset (sklearn Implementation)")
plt.show()
```

```
Shape of Data: (150, 4)
Shape of transformed Data: (150, 2)
```

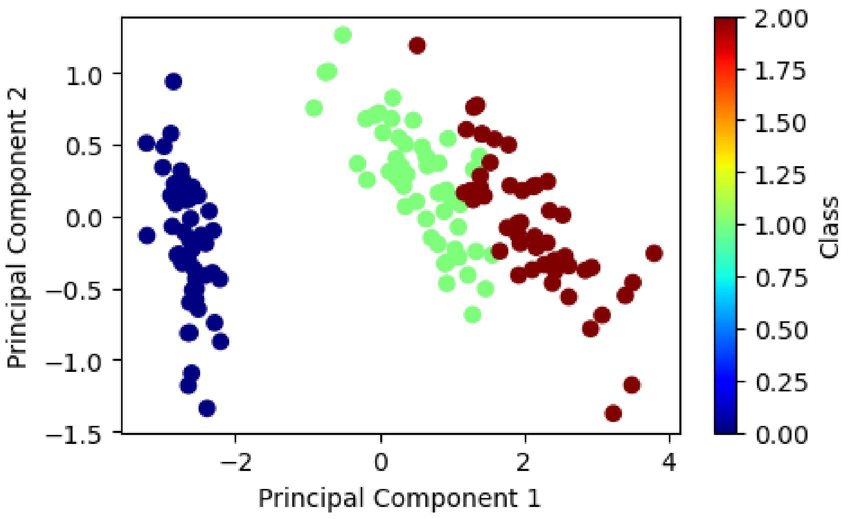


PCA Without sklearn

In [5]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris

class PCA:
    def fit_transform(self, X, n_components=2):
        # Mean center the data
        mean = np.mean(X, axis=0)
        X_centered = X - mean

        # Calculate covariance matrix
        cov = np.cov(X_centered.T)

        # Calculate eigenvalues and eigenvectors
        eigenvalues, eigenvectors = np.linalg.eig(cov)

        # Sort the eigenvectors in decreasing order of eigenvalues
        idxs = np.argsort(eigenvalues)[::-1]
        eigenvectors = eigenvectors[:, idxs]

        # Select the top n_components eigenvectors
        components = eigenvectors[:, :n_components]

        # Transform the data
        X_projected = np.dot(X_centered, components)
        return X_projected

# Load dataset
X = load_iris().data
y = load_iris().target

# Perform PCA
pca = PCA()
X_projected = pca.fit_transform(X)

print("Shape of Data:", X.shape)
print("Shape of transformed Data:", X_projected.shape)

# Plot the results
plt.figure(figsize = (5,3))
plt.scatter(X_projected[:, 0], X_projected[:, 1], c=y, cmap="jet")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.colorbar(label='Class')
plt.show()
```

```
Shape of Data: (150, 4)
Shape of transformed Data: (150, 2)
```


Principal Component 2
Principal Component 1
Class