PK1

```
class CD:
   def __init__(self, cd_id, title, artist, genre, minutes):
      self.cd_id = cd_id
      self.title = title
      self.artist = artist
      self.genre = genre
      self.duration = minutes
   def __repr__(self):
      return f"CD(ID: {self.cd_id}, Title: '{self.title}', Artist: '{self.artist}', Genre:
'{self.genre}', Minutes: '{self.duration}') \n"

class Shop:
   def __init__(self, library_id, name):
      self.library_id = library_id
      self.name = name
      self.cds = []
   def add_cd(self, cd):
      self.cds.append(cd)
   def __repr__(self):
      return f"Library(ID: {self.library_id}, Name: '{self.name}', CDs: {self.cds}) \n "

class CDLibrary:
   def __init__(self):
      self.relationships = []
   def add_relationship(self, cd, library):
      self.relationships.append((cd, library))
      library.add_cd(cd)
   def __repr__(self):
      return f"CDLibrary(Relationships: {len(self.relationships)}) \n"
   def list_cds_in_libraries(self):
      result = {}
      for cd, library in self.relationships:
         if library.name not in result:
            result[library.name] = []
         result[library.name].append(cd)
      return result
   def count_total_duration_per_library(self):
      duration_count = {}
      for cd, library in self.relationships:
         if library.name not in duration_count:
            duration_count[library.name] = 0
         duration_count[library.name] += cd.duration
      return sorted(duration_count.items(), key=lambda x: x[1], reverse=True)
   def list_libraries_with_cd_in_name(self, name_part):
      result = {}
      for cd, library in self.relationships:
         if name_part in library.name:
            if library.name not in result:
               result[library.name] = []
            result[library.name].append(cd)
      return result
```

```
if __name__ == "__main__":
    cd1 = CD(1, "Thriller", "Michael Jackson", "Pop", 40)
    cd2 = CD(2, "Back in Black", "AC/DC", "Rock", 35)
    cd3 = CD(3, "The Dark Side of the Moon", "Pink Floyd", "Rock", 123)
    cd4 = CD(4, "Kiss Of Death", "Motorhead", "Metal", 51)
    cd5= CD(5, "Alison Hell", "Annihilator", "Metal", 44)
    cd6 = CD(6, "The Number Of The Beast", "Iron Maiden", "Metal", 55)
    library1 = Shop(1, "Pop mania")
    library2 = Shop(2, "Rock House")
    library3 = Shop(3,  "Metall Invaders")

    cd_library = CDLibrary()

    cd_library.add_relationship(cd1, library1)
    cd_library.add_relationship(cd2, library1)
    cd_library.add_relationship(cd3, library2)
    cd_library.add_relationship(cd4, library3)
    cd_library.add_relationship(cd5, library3)
    cd_library.add_relationship(cd6, library3)
    cd_library.add_relationship(cd6, library2)
    cd_library.add_relationship(cd4, library1)
    cd_library.add_relationship(cd1, library2)
    cd_library.add_relationship(cd3, library3)

    # Вывод всех связанных дисков и магазинов
    print("CDs in Shops:")
    for library_name, cds in cd_library.list_cds_in_libraries().items():
        print(f"{library_name}: {cds}")

    # Подсчет длительности дисков в каждом магазине и вывод
    print("Count of Rock CDs in Shops:")
    for library_name, count in cd_library.count_total_duration_per_library():
        print(f"{library_name}: {count} minutes")

    # Вывод магазинов, в названии которых присутствует слово 'House' и их содержимого
    print("Shops containing 'House' and their names:")
    for library_name, cds in cd_library.list_libraries_with_cd_in_name('House').items():
        print(f"{library_name}: {cds}")
```

РК2 1 тест

```
#Проверяем, работают ли методы так, как ожидается
import unittest
class CD:
    def __init__(self, cd_id, title, artist, genre, minutes):
        self.cd_id = cd_id
        self.title = title
        self.artist = artist
        self.genre = genre
        self.duration = minutes

    def __repr__(self):
```

```
        return f"CD(ID: {self.cd_id}, Title: '{self.title}', Artist: '{self.artist}', Genre:
'{self.genre}', Minutes: '{self.duration}')"


class Shop:
    def __init__(self, library_id, name):
        self.library_id = library_id
        self.name = name
        self.cds = []

    def add_cd(self, cd):
        self.cds.append(cd)

    def __repr__(self):
        return f"Library(ID: {self.library_id}, Name: '{self.name}', CDs: {self.cds})"


class CDLibrary:
    def __init__(self):
        self.relationships = []

    def add_relationship(self, cd, library):
        self.relationships.append((cd, library))
        library.add_cd(cd)

    def __repr__(self):
        return f"CDLibrary(Relationships: {len(self.relationships)})"

    def list_cds_in_libraries(self):
        result = {}
        for cd, library in self.relationships:
            if library.name not in result:
                result[library.name] = []
            result[library.name].append(cd)
        return result

    def count_total_duration_per_library(self):
        duration_count = {}
        for cd, library in self.relationships:
            if library.name not in duration_count:
                duration_count[library.name] = 0
            duration_count[library.name] += cd.duration
        return sorted(duration_count.items(), key=lambda x: x[1], reverse=True)

    def list_libraries_with_cd_in_name(self, name_part):
        result = {}
        for cd, library in self.relationships:
            if name_part in library.name:
                if library.name not in result:
                    result[library.name] = []
                result[library.name].append(cd)
        return result
```

```
class TestCDLibrary(unittest.TestCase):

    def setUp(self):
        self.cd1 = CD(1, "Thriller", "Michael Jackson", "Pop", 40)
        self.cd2 = CD(2, "Back in Black", "AC/DC", "Rock", 35)
        self.cd3 = CD(3, "The Dark Side of the Moon", "Pink Floyd", "Rock", 123)
        self.library1 = Shop(1, "Pop mania")
        self.library2 = Shop(2, "Rock House")
        self.cd_library = CDLibrary()

    def test_cd_initialization(self):
        self.assertEqual(self.cd1.cd_id, 1)
        self.assertEqual(self.cd1.title, "Thriller")
        self.assertEqual(self.cd1.artist, "Michael Jackson")
        self.assertEqual(self.cd1.genre, "Pop")
        self.assertEqual(self.cd1.duration, 40)

    def test_shop_initialization(self):
        self.assertEqual(self.library1.library_id, 1)
        self.assertEqual(self.library1.name, "Pop mania")
        self.assertEqual(len(self.library1.cds), 0)

    def test_add_cd_to_shop(self):
        self.library1.add_cd(self.cd1)
        self.assertEqual(len(self.library1.cds), 1)
        self.assertIn(self.cd1, self.library1.cds)

    def test_add_relationship(self):
        self.cd_library.add_relationship(self.cd1, self.library1)
        self.assertIn((self.cd1, self.library1), self.cd_library.relationships)
        self.assertIn(self.cd1, self.library1.cds)

    def test_list_cds_in_libraries(self):
        self.cd_library.add_relationship(self.cd1, self.library1)
        self.cd_library.add_relationship(self.cd2, self.library2)
        result = self.cd_library.list_cds_in_libraries()
        self.assertEqual(len(result), 2)
        self.assertIn(self.library1.name, result)
        self.assertIn(self.library2.name, result)
        self.assertIn(self.cd1, result[self.library1.name])
        self.assertIn(self.cd2, result[self.library2.name])

    def test_count_total_duration_per_library(self):
        self.cd_library.add_relationship(self.cd1, self.library1)
        self.cd_library.add_relationship(self.cd2, self.library2)
        result = self.cd_library.count_total_duration_per_library()
        self.assertEqual(result[0][0], self.library1.name)
        self.assertEqual(result[0][1], 40)
        self.assertEqual(result[1][0], self.library2.name)
        self.assertEqual(result[1][1], 35)

    def test_list_libraries_with_cd_in_name(self):
```

```
        self.cd_library.add_relationship(self.cd1, self.library1)
        self.cd_library.add_relationship(self.cd2, self.library2)
        result = self.cd_library.list_libraries_with_cd_in_name("House")
        self.assertIn(self.library2.name, result)
        self.assertNotIn(self.library1.name, result)

if __name__ == "__main__":
    unittest.main()
```

РК2 2 тест

```
import unittest
#проверка взаимодействия классов
class CD:
    def __init__(self, cd_id, title, artist, genre, minutes):
        self.cd_id = cd_id
        self.title = title
        self.artist = artist
        self.genre = genre
        self.duration = minutes

    def __repr__(self):
        return f"CD(ID: {self.cd_id}, Title: '{self.title}', Artist: '{self.artist}', Genre:
'{self.genre}', Minutes: '{self.duration}')"

class Shop:
    def __init__(self, library_id, name):
        self.library_id = library_id
        self.name = name
        self.cds = []

    def add_cd(self, cd):
        self.cds.append(cd)

    def __repr__(self):
        return f"Library(ID: {self.library_id}, Name: '{self.name}', CDs: {self.cds})"

class CDLibrary:
    def __init__(self):
        self.relationships = []

    def add_relationship(self, cd, library):
        self.relationships.append((cd, library))
        library.add_cd(cd)

    def __repr__(self):
        return f"CDLibrary(Relationships: {len(self.relationships)})"

    def list_cds_in_libraries(self):
        result = {}
        for cd, library in self.relationships:
            if library.name not in result:
                result[library.name] = []
            result[library.name].append(cd)
```

```
        return result

    def count_total_duration_per_library(self):
        duration_count = {}
        for cd, library in self.relationships:
            if library.name not in duration_count:
                duration_count[library.name] = 0
            duration_count[library.name] += cd.duration
        return sorted(duration_count.items(), key=lambda x: x[1], reverse=True)

    def list_libraries_with_cd_in_name(self, name_part):
        result = {}
        for cd, library in self.relationships:
            if name_part in library.name:
                if library.name not in result:
                    result[library.name] = []
                result[library.name].append(cd)
        return result

class TestCDLibraryAdditional(unittest.TestCase):

    def setUp(self):
        self.cd1 = CD(1, "Thriller", "Michael Jackson", "Pop", 40)
        self.cd2 = CD(2, "Back in Black", "AC/DC", "Rock", 35)
        self.cd3 = CD(3, "The Dark Side of the Moon", "Pink Floyd", "Rock", 123)
        self.cd4 = CD(4, "Kiss Of Death", "Motorhead", "Metal", 51)
        self.cd5 = CD(5, "Alison Hell", "Annihilator", "Metal", 44)
        self.cd6 = CD(6, "The Number Of The Beast", "Iron Maiden", "Metal", 55)

        self.library1 = Shop(1, "Pop mania")
        self.library2 = Shop(2, "Rock House")
        self.library3 = Shop(3, "Metall Invaders")

        self.cd_library = CDLibrary()
        self.cd_library.add_relationship(self.cd1, self.library1)
        self.cd_library.add_relationship(self.cd2, self.library1)
        self.cd_library.add_relationship(self.cd3, self.library2)
        self.cd_library.add_relationship(self.cd4, self.library3)
        self.cd_library.add_relationship(self.cd5, self.library3)
        self.cd_library.add_relationship(self.cd6, self.library3)
        self.cd_library.add_relationship(self.cd6, self.library2)
        self.cd_library.add_relationship(self.cd4, self.library1)
        self.cd_library.add_relationship(self.cd1, self.library2)
        self.cd_library.add_relationship(self.cd3, self.library3)

    def test_add_relationship(self):
        new_cd = CD(7, "New Album", "New Artist", "New Genre", 60)
        new_library = Shop(4, "New Library")

        self.cd_library.add_relationship(new_cd, new_library)
        self.assertIn(new_cd, new_library.cds)
        self.assertIn((new_cd, new_library), self.cd_library.relationships)
```

```
    def test_count_total_duration_empty_library(self):
        empty_library = Shop(5, "Empty Library")
        empty_cd_library = CDLibrary()
        result = empty_cd_library.count_total_duration_per_library()
        expected = []
        self.assertEqual(result, expected)



if __name__ == "__main__":
    unittest.main()
```

РК2 3 тест

```
import unittest
#Проверка добавления
class CD:
    def __init__(self, cd_id, title, artist, genre, minutes):
        self.cd_id = cd_id
        self.title = title
        self.artist = artist
        self.genre = genre
        self.duration = minutes

    def __repr__(self):
        return f"CD(ID: {self.cd_id}, Title: '{self.title}', Artist: '{self.artist}', Genre:
'{self.genre}', Minutes: {self.duration})"


class Shop:
    def __init__(self, library_id, name):
        self.library_id = library_id
        self.name = name
        self.cds = []

    def add_cd(self, cd):
        self.cds.append(cd)

    def __repr__(self):
        return f"Library(ID: {self.library_id}, Name: '{self.name}', CDs: {self.cds})"


class CDLibrary:
    def __init__(self):
        self.relationships = []

    def add_relationship(self, cd, library):
        self.relationships.append((cd, library))
        library.add_cd(cd)

    def __repr__(self):
```

```
        return f"CDLibrary(Relationships: {len(self.relationships)})"

    def list_cds_in_libraries(self):
        result = {}
        for cd, library in self.relationships:
            if library.name not in result:
                result[library.name] = []
            result[library.name].append(cd)
        return result

    def count_total_duration_per_library(self):
        duration_count = {}
        for cd, library in self.relationships:
            if library.name not in duration_count:
                duration_count[library.name] = 0
            duration_count[library.name] += cd.duration
        return sorted(duration_count.items(), key=lambda x: x[1], reverse=True)

    def list_libraries_with_cd_in_name(self, name_part):
        result = {}
        for cd, library in self.relationships:
            if name_part in library.name:
                if library.name not in result:
                    result[library.name] = []
                result[library.name].append(cd)
        return result


class TestCDLibrary(unittest.TestCase):
    def setUp(self):
        self.cd1 = CD(1, "Pop Hits", "Various Artists", "Pop", 45)
        self.cd2 = CD(2, "Rock Classics", "Various Artists", "Rock", 50)
        self.cd3 = CD(3, "House Anthems", "DJ Artist", "House", 48)
        self.cd4 = CD(4, "Pop mania", "Pop Artist", "Pop", 40)
        self.cd5 = CD(5, "Heavy Metal", "Annihilator", "Metal", 52)
        self.cd6 = CD(6, "Iron Legends", "Iron Maiden", "Metal", 60)

        self.library1 = Shop(1, "Pop mania")
        self.library2 = Shop(2, "Rock House")
        self.library3 = Shop(3, "Metall Invaders")

        self.cd_library = CDLibrary()
        self.cd_library.add_relationship(self.cd1, self.library1)
        self.cd_library.add_relationship(self.cd2, self.library2)
        self.cd_library.add_relationship(self.cd4, self.library1)

    def test_list_libraries_with_cd_in_name_multiple_matches(self):
        self.cd_library.add_relationship(self.cd3, self.library2)  # DJ Artist in Rock House
        self.cd_library.add_relationship(self.cd5, self.library3)  # Annihilator in Metall Invaders
        self.cd_library.add_relationship(self.cd6, self.library3)  # Iron Maiden in Metall Invaders

        # Check libraries with "House"
```

```
        result = self.cd_library.list_libraries_with_cd_in_name("House")
        expected = {
            "Rock House": [self.cd2, self.cd3]
        }
        self.assertEqual(result, expected)

        # Check libraries with "Invaders"
        result = self.cd_library.list_libraries_with_cd_in_name("Invaders")
        expected = {
            "Metall Invaders": [self.cd5, self.cd6]
        }
        self.assertEqual(result, expected)

        # Check libraries with "Pop"
        result = self.cd_library.list_libraries_with_cd_in_name("Pop")
        expected = {
            "Pop mania": [self.cd1, self.cd4]
        }
        self.assertEqual(result, expected)

if __name__ == "__main__":
    unittest.main()
```