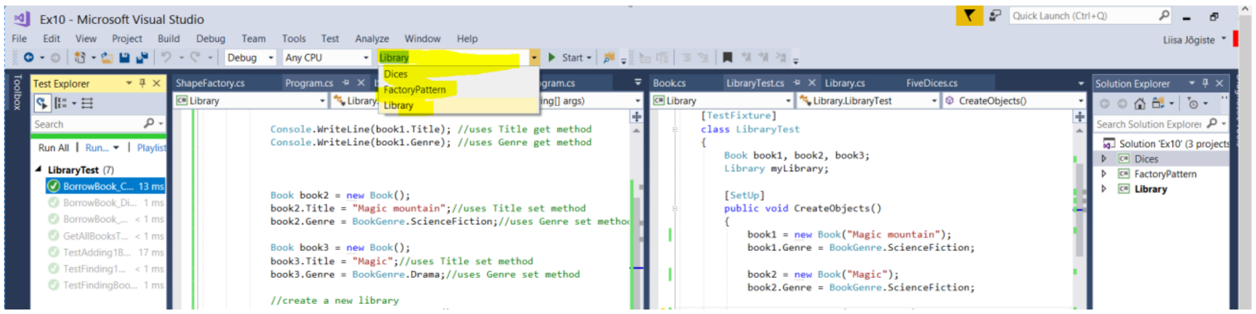
NB! Uploaded exercise solution in ained.ttu.ee has multiple sub projects to keep things separately. You can start different projects by changing startup project:



```
            Console.WriteLine(book1.Title); //uses Title get method
            Console.WriteLine(book1.Genre); //uses Genre get method



            Book book2 = new Book();
            book2.Title = "Magic mountain";//uses Title set method
            book2.Genre = BookGenre.ScienceFiction;//uses Genre set metho

            Book book3 = new Book();
            book3.Title = "Magic";//uses Title set method
            book3.Genre = BookGenre.Drama;//uses Genre set method

            //create a new library
```

```
    [TestFixture]
    class LibraryTest
    {
        Book book1, book2, book3;
        Library myLibrary;

        [SetUp]
        public void CreateObjects()
        {
            book1 = new Book("Magic mountain");
            book1.Genre = BookGenre.ScienceFiction;

            book2 = new Book("Magic");
            book2.Genre = BookGenre.ScienceFiction;
```

Exercise 1: Factory pattern

https://csharpdesignpatterns.codeplex.com/wikipage?title=Factory%20Method%20Pattern&referringTitle=Home

1. Create an interface for dog with method Speak()
2. Create 3 dog class objects:
    a. Poodle, method Speak content is: Console.WriteLine("The poodle says \"arf\"");
    b. Rottweiler, method Speak content is: Console.WriteLine("The Rottweiler says (in a very deep voice) \"WOOF!\"");
    c. SiberianHusky, method Speak content is: Console.WriteLine("The husky says \"Dude, what's up?\"");
3. Create factory with method GetDog(string criteria) which returns concrete dog object:
    a. Criteria „small" - > create poodle
    b. Criteria „big"-> create Rottweiler
    c. Criteria „working"->create SiberianHusky
    d. Criteria smth else: return null

Method syntax: *public static Dog getDog(String criteria)*

4. In main class create all dog objects through factory and call out the speak method. Also test the null case.

NB! Follow the factory pattern; in main class use only interface class name, not Poodle or Rottwiler or SiberianHusky.

Exercise 2: Improve the book-library solution. Take the last version from ained.ttu.ee page.

1. Create logic for returning the book (*complete the „ReturnBook()“ method*)
    The book can only be returned if we have a book with the same name and status borrowed out.
    Example: Library.ReturnBook(„SomeName“)
    Returning changes the field value of *_isBorrowed* to false. (See how BorrowBook() is done)
2. Add an author for the book. Use get/set to add the author. Add an additional constructor for book which takes the author name as a second parameter in addition to title.

Example: *Book myBook = new Book(„Blindness“, „Jose Saramago“))*

3. Add release year for the book. Use get/set.
   Example: *book1.ReleaseYear = 1765*
4. Add a name and address for the libray. Use get/set for both. Add an additional constructor for library which takes the librarys name as a parameter.
   Example: *Library library = new Library(„Town library“)*
5. Add a method for printing librarys info (name and address and the number of books).
   Example: libray.PrintInfo() -> „Library with name x is located in y and has 5 books“
6. Add method for searching books by authors name from library
   Example: *List<Books> booksBySaramago = library.FindBooksByAuthor(„Saramago“);*
7. Add method for comparing two book objects in Book class. Two book objects are equal if their title and autor is equal.
   Example: *bool areBooksEqual = book1.CompareBook(book2);*
8. Add method for printing out book info: title, author and release year.
   Example: *book1.PrintInfo() –> „Magic Mountain by Thomas Mann, 1924“*
9. Use the method from previous task and update the existing methods:
   `PrintAllBooksByName()` and `PrintAllBookTitlesInLibrary()`.
   Instead of only the title, the whole info about book should be printed (title, author and year).
10. Add a method for printing out all the books by a certain author. Use the method created in pt 6 for getting all the books. Use print method from pt 8.
    Example: *library.PrintAllBooksByAuthor("Saramago"). -> Prints out all books by this author*
    Tip: check method `PrintAllBooksByName()`
11. Add a method for getting all the books from library that are written after a certain year.
    Example: *List<Book> booksReleasedAfter1900 = library.GetBooksReleasedAfterYear(1900)*
12. Add one new genre value for books (Choose the value yourself). Create at least 1 new book with this genre.

For better understanding test the methods both in main and test class. (Void methods for printing info are not needed to be tested in test class. They should be called out in main method.)

Create at least 10 books for testing puropses!

Tips:

- Think of real world examples.
- Look at the existing methods, all the solutions you need are there!
- In some cases you might need to check if objects property is present. You can do this: book1.Year !=null
- With tests you can check different values:
  - Length of the list (for example if you search for books with certain genre you know what is the expected result).
  - Titles of objects
  - Borrowing status
  - etc