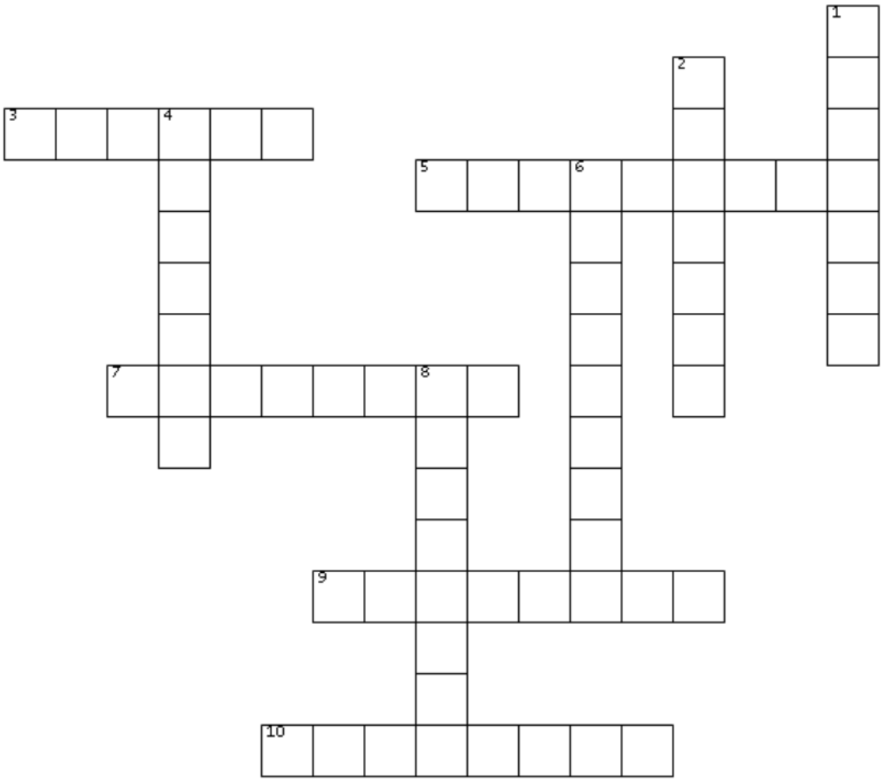
Introduction to Software Engineering
Assignment 10

Walter Tichy

Title page: Create a title page with ”Introduction to Software Engineering”, “Assignment 10”, your name, and date of completion.

Problem 1 (2 pt): Crossword puzzle for design patterns. Test yourself by doing this from memory. Do not look at the lecture slides!



ACROSS
3. Provides a uniform interface to a subsystem's set of interfaces to simplify the use of the subsystem.
5. Allows clients to treat both individual objects and composites of objects uniformly.
7. Promotes loose coupling by preventing objects from explicitly referencing each other.
9. Algorithms can vary independent of the clients that use them.
10. Allows sequential access to the elements of a compound object without revealing its underlying representation.
DOWN
1. Capture and externalize an object's internal state without violating its encapsulation, so that the object can later be restored to that state.
2. Allows to define a new operation without modifying the classes of the elements it operates on.
4. Converts a class's interface to another interface clients expect.

6. Determines the types of objects to be created using a typical instance and creates new objects by copying this instance.
8. Defines a 1-to-n dependency between objects, such that changing the object's state results in all dependent objects being notified.

Problem 2 (5 pt): JUnit 5. Given the following method signature with the corresponding JavaDoc comment of a method that returns an element at a specific location in a sorted list:

```
public class SortedIntegerList {
  /**
   * Returns the element at the given position in the list.
   *
   * @param position
   *            the position of the list element to be returned.
   * @return    the element at the given position in the list.
   *
   * @throws IndexOutOfBoundsException
   *            the exception is thrown if the given position
   *            is outside the bounds of the list.
   */
  public int getElement(int position)
    throws IndexOutOfBoundsException;
}
```

Complete the JUnit test class below. The class should provide useful test cases for equivalence classes and boundary values. Hint: The list contains 100 elements. Indices run from 0 to 99. The constructor initializes the list with the integers 0 to 99 in ascending order. Find out how to check for exceptions in JUnit 5 on the Internet.

```
//imports
public class ListTest {
  static SortedIntegerList list;

  @BeforeAll
  public static void setup() {
    //allocate list with 100 elements with values 0..99
    list = new SortedIntegerList(100);
  }
```

Problem 3 (10 pt): Git. Which git commands perform the following series of actions.
Hint: if you need help read the relevant chapters of this book: https://git-scm.com/book/en/v2

1. The git repository `myproject` is stored on a server under the URL `https://github.com/myproject`. Which command downloads the entire repository, including the latest version of the software?
2. You create a new file `f1` in your local directory `myproject`. Which command tells git to start tracking `f1`?
3. You modify an existing file `f2` in `myproject`. How do you tell git to include this new version in the next snapshot? (Do not yet create that snapshot.)
4. There is a file `f3` that is no longer needed in the next snapshot. How do you tell git to get rid of `f3` in the next snapshot, but not in the preceding ones?
5. Now create a new snapshot with a log message `“f1 new, f2 changed, f3 deleted”.`
6. Next, create a branch called `testing`.
7. Switch to that branch.
8. You change file `f2` again with an editor. How do you create a new snapshot on the branch `testing` including the new version of `f2`?
9. You find out that there is a branch called `hotfix.` You need to merge `testing` with `hotfix`. How is that done?
10. Finally, how do you upload the master branch to the server?