# Lab: SOLID

This document defines the exercises for ["C# OOP Advanced" course @ Software University](https://softuni.bg/courses/csharp-oop-advanced-high-quality-code).

## Stream Progress Info

Refactor code for this task, so **Stream Progress Info** can work with different kinds of **Streams**. First make sure it works with **Music** too. Refactor code, so in the future if a **new kind of stream** is introduced, you will need **just to import one new class** with **BytesSent** and **Length** getters in it.

## Graphic Editor

Refactor code for this task, so **Graphic Editor can draw all kind of shapes** without checking, **what kind is concrete shape.** In the future new shapes will be added to system, so prepare the system for this moments. When you **add new shape**, you just should **add new class and nothing more**.

## Detail Printer

Refactor code for this task, so **Detail Printer** don’t need to ask **what kind of employee is passed to it**. Detail Printer need just to print details for all kind of employees. When new kind of employee is added you will need just to **add new class and nothing else.**

## Recharge

You are given a library with the following classes

* Worker implements ISleeper
* Employee inherits Worker
* Robot inherits Worker
* RechargeStation

If you inspect the code, you can see that some of the classes have methods that they can't use (throw UnsupportedOpperationException) which is clear indication that the code should be refactored.

Refactor the structure so that it conforms to the **Interface Segregation** principle.

### Hints

Make the Robot to extend Worker and at the same time to implement Rechargeable



## Security Door

You are given:

* SecurityManager
* abstract class SecurityCheck
* interface SecurityUI

SecurityManager which can interact with a user by validating his key card or by getting his pin code. Both methods are provided by an interface called SecurityUI. The validation is performed by the appropriate SecurityCheck class.

Refactor the structure so that it conforms to the **Interface Segregation** principle.

### Hints

Split SecurityUI into smaller role interfaces, one for each SecurityCheck class

