AGH University of Krakow | Faculty of Mechanical Engineering and Robotics
| Department of Robotics and Mechatronics

MECHATRONIC ENGINEERING | I cycle | semester VI | 2025/2026

Object oriented programming and software engineering

Instruction XII:

Design thinking

You will learn:

How to improve software systems by focusing on usability and user interaction instead of only implementation. You will learn how to analyze a system from the user perspective, identify usability problems, simplify workflows and interaction, improve applications and redesign poorly structured program behavior.

By the end of the class, you should have a better understanding of how software is experienced by users and how interaction design affects usability.

Course supervisor:
dr inż. Lucjan Miękina, miekina@agh.edu.pl

Instruction author:
mgr inż. Joanna Koszyk, jkoszyk@agh.edu.pl

1. Initial information

A program can work correctly and still be difficult or frustrating to use. Many software problems are not caused by incorrect code, but by poor design decisions such as too many user actions, confusing menus, unclear information, poor navigation, unnecessary complexity.

Even terminal applications should be easy to understand and operate.

A user should quickly understand what the program does, what options are available, what input is expected, what happened after an action was performed.

Design thinking focuses on understanding the user and improving how the system behaves from the user perspective. Instead of asking “how do we implement this?”, design thinking asks “how should the user interact with this system?”.

2. Theoretical Content

Example of poor design:

```
#include <iostream>

int main(){
    int option;

    while(true){
        std::cout << "1. Open menu\n";
        std::cin >> option;

        if(option == 1){
            std::cout << "1. Enter parking\n";
            std::cout << "2. Exit parking\n";
            std::cin >> option;

            if(option == 1){
                char confirm;
                std::cout << "Confirm entry? y/n\n";
                std::cin >> confirm;

                if(confirm == 'y'){
                    std::cout << "Vehicle entered\n";
                }
            }
        }
    }
}
```

Improved version:

```
#include <iostream>

int main(){
    int option;

    while(true){
        std::cout << "\n=== Parking System ===\n";
        std::cout << "1. Enter parking\n";
        std::cout << "2. Exit parking\n";
        std::cout << "3. Exit program\n";
```

```
            std::cout << "Choose option: ";

            std::cin >> option;

            switch(option){
                case 1:
                    std::cout << "Vehicle entered\n";
                    break;

                case 2:
                    std::cout << "Vehicle exited\n";
                    break;

                case 3:
                    return 0;

                default:
                    std::cout << "Invalid option\n";
            }
        }

        return 0;
    }
```

3. Tasks

TASK 1.

You are given a washing machine controller code. Analyze the interaction with user and improve the code.

```
    int option;
    char confirm;

    std::cout << "1. Start menu\n";
    std::cin >> option;

    if(option == 1){

        std::cout << "1. Cotton\n";
        std::cout << "2. Synthetic\n";
        std::cin >> option;

        if(option == 1){

            std::cout << "Cotton selected\n";

            std::cout << "Continue? y/n\n";
            std::cin >> confirm;

            if(confirm == 'y'){

                std::cout << "Select temperature:\n";
                std::cout << "1. 40\n";
                std::cout << "2. 60\n";

                std::cin >> option;

                std::cout << "Program started\n";
            }
        }
    }
```

TASK 2.

You are given an elevator controller code. Improve the usability.

```
int option;

while(true){

    std::cout << "1. Elevator system\n";
    std::cin >> option;

    if(option == 1){

        std::cout << "1. Floor 1\n";
        std::cout << "2. Floor 2\n";
        std::cout << "3. Floor 3\n";

        std::cin >> option;

        if(option == 2){

            std::cout << "Selected floor 2\n";

            std::cout << "Press 1 to confirm\n";
            std::cin >> option;

            if(option == 1){

                std::cout << "Moving...\n";

                std::cout << "Press 1 to open door\n";
                std::cin >> option;

                if(option == 1){
                    std::cout << "Door opened\n";
                }
            }
        }
    }
}
```

TASK 3.

You are given a bike rental system code containing unclear interaction and poor user feedback.

```
int bike;
int payment;

std::cout << "Bike?\n";
std::cin >> bike;

std::cout << "Money?\n";
std::cin >> payment;

if(payment >= 10){
    std::cout << "OK\n";
}
else{
    std::cout << "Wrong\n";
}
```

TASK 4.

Change seats and work with another team member from a different mini-project group. Without receiving any explanation, try to use the program only based on the displayed menus, messages, and interaction flow.

While testing the application, try to understand how the system works, perform different actions, enter incorrect or unexpected input and navigate through all available options. Identify where the interaction becomes confusing or unclear.

After returning to your team, redesign the interaction based on the received feedback.