Tutoriat 6
Abstractizare

1. Ce este?
   Ascunderea detaliilor implementării față de utilizatorul unei clase.

2. Exemple din practică
   Ex1: Când includem o bibliotecă, folosim funcțiile/metodele din acea bibliotecă, fără a fi nevoie să cunoaștem implementarea acestora.

   Ex2: Când scriem o clasă într-un fișier de tip header(.h), în programul principal doar apelăm metodele publice, fără a fi nevoie să cunoaștem implementarea acestora.

3. Metode pur virtuale
   Sunt metode virtuale fără implementare.

```
virtual void nume_metodă() = 0;
```

4. Clase abstracte
   Clase care conțin **cel puțin o metodă pur virtuală**.

   Obs:
   - **Nu** pot fi instanțiate.
   - Clasele lor derivate pot fi instanțiate doar dacă au fost **implementate toate metodele pur virtuale.**

5. Clasă abstractă vs interfață(în C++):
   Interfața are **doar metode pur virtuale**, spre deosebire de clasele abstracte care pot avea și alte tipuri de metode, pe lângă cele pur virtuale.

```
class MyInterface
{
public:
    // Empty virtual destructor for proper cleanup
    virtual ~MyInterface() {}

    virtual void Method1() = 0;
    virtual void Method2() = 0;
};


class MyAbstractClass
```

```
{
public:
    virtual ~MyAbstractClass();

    virtual void Method1();
    virtual void Method2();
    void Method3();

    virtual void Method4() = 0; // make MyAbstractClass not instantiable
};
```

6. Exemplu abstractizare

```
class Animal {
public:
    virtual void eat() = 0;
    virtual void sleep() = 0;
};
class Dog : public Animal {
public:
    void eat() {
        cout << "Dog::eat()";
    }
    void sleep() {
        cout << "Dog::sleep()";
    }
    void bark() {
        cout << "Dog::bark()";
    }
};
class Cat : public Animal {
public:
    void eat() {
        cout << "Cat::eat()";
    }
    void sleep() {
        cout << "Cat::sleep()";
    }
    void meow() {
        cout << "Cat::meow()";
    }
};
int main() {
    Animal a; // eroare de compilare => Animal este clasa abstracta=> nu poate fi instantiata
    Dog d; // corect
    Cat c; // corect
    Animal* a1 = new Dog; // corect (se declara un pointer de tipul clasei abstracte, dar se intasntiaza un obiect in memorie de tipul clasei neabstracte)
```

```
    Animal* a2 = new Cat; // corect
    return 0;
}
```