PCLP2

Laboratorul 7

1. **Moştenirea claselor. Clasa `Employee` şi clasa `HourlyWorker`.** Moştenirea (derivarea) claselor este o metodă prin care se pot implementa clase noi refolosind clase deja existente. Clasele derivate moştenesc datele membre şi funcţiile membre ale claselor de bază. În clasele derivate se pot adăuga date membre şi funcţii membre noi, se pot supraîncărca funcţii moştenite sau se pot suprascrie unele funcţii moştenite. Două funcţii sunt supraîncărcate dacă au acelaşi nume dar semnături diferite. O funcţie suprascrie (rescrie) o altă funcţie dacă cele două funcţii au acelaşi nume şi aceeaşi semnătură. Testaţi programul de mai jos. Clasa `HourlyWorker` este derivată din clasa `Employee` şi preia cele două date membre `firstName` şi `lastName`, cărora le adaugă datele membre `wage` si `hours`. Funcţia `print()` este implementată în clasa `Employee` şi este suprascrisă în clasa `HourlyWorker`.

Employee.h

```
#ifndef EMPLOYEE_H
#define EMPLOYEE_H
#include <iostream>
using namespace std;
class Employee
{
  public:
    Employee();
    Employee(string, string);
    void print();
  protected:
    string firstName;
    string lastName;
};
#endif
```

Employee.cpp

```
#include "Employee.h"
Employee::Employee()
{
  firstName = "";
  lastName = "";
}
Employee::Employee(string fn, string ln)
{
  firstName = fn;
  lastName = ln;
}
void Employee::print()
{
  cout << "Se executa Employee::print()" << endl;
  cout << "--Angajatul " << firstName << " "
                         << lastName << endl;
}
```

HourlyWorker.h

```
#ifndef HOURLY_WORKER_H
#define HOURLY_WORKER_H
#include "Employee.h"
#include <iostream>
using namespace std;
class HourlyWorker : public Employee
{
  public:
    HourlyWorker(string, string, double, double);
    double getPay();
    void print();
  private:
    double wage;
    double hours;
};
#endif
```

HourlyWorker.cpp

```
#include "HourlyWorker.h"
#include <iomanip>
HourlyWorker::HourlyWorker(string fn, string ln,
                           double w, double h)
{
  firstName = fn;
  lastName =ln;
  wage = w;
  hours = h;
}
void HourlyWorker::print()
{
  cout << endl << "Se executa HourlyWorker::print()"
       << endl;
  Employee::print();
  cout << "Se revine la functia HourlyWorker::print()"
       << endl;
  cout << "--este platit cu "
       << setiosflags(ios::fixed | ios::showpoint)
       << setprecision(2) << getPay() << " Lei" << endl;
}
double HourlyWorker::getPay()
{
  return wage * hours;
}
```

main.cpp

```
#include "HourlyWorker.h"
using namespace std;
int main()
{
  Employee e1("Gheorghe", "Popescu");
  e1.print();
  HourlyWorker e2("Vasile", "Ionescu", 10, 9);
```

```
    e2.print();
    return 0;
}
```

2. Urmând exemplul de mai sus, implementaţi clasa `Time` şi clasa `DateTime` derivată din `Time`. Folosiţi definiţia clasei `Time` din laboratorul 3. Clasa `DateTime` adaugă datele membre `day`, `month`, `year` conform modelului din laboratorul 4. Cele două clase implementează funcţia `print()`. Scrieţi o variantă a funcţiei `main` care declară şi tipăreşte un obiect din clasa `Time` şi un obiect din clasa `DateTime`.