Test Examen POO

Petculescu Mihai-Silviu

1.1

Rezultat

```
1
2
The End
```

Folosind ca si referinţă variabila statică sem, comuna tuturol instanţelor clasei Fir, prin linia "synchronized(sem)" ne asigurăm ca blocul de instrucţiuni aferent ei este "thread protected", în maniera în care doar un tred, la un singur moment în timp, poate avea acces la acesta (read/write). Astfel thredul alocat instanţei f1 porneşte primul şi devine "monitor" al acelui bloc de cod, iar după ce acesta îşi termină execuţia, elibereaza variabila sem şi urmează execuţia thredului alocat instanţei f2.

1.2

Rezultat

```
Oricare din combinatiile 1 1 ; 1 2 ; 2 1 ; 2 2 The End
Depinde de disponibilitatea procesorului la momentul executiei.
```

Eliminând linia "synchr_sem" blocul de cod nu mai este thread protected şi in consecinţă variabila statică x poate fi accesată şi modificată de ambele threaduri, fără restricţii şi cu posibile (chiar destul de probabile) execuţiilor suprapuse. Astfel threadul alocat instanţei poate să termine înaintea celui alocat instanţei f1, deci să se afişeze mai rapid, sau chiar sa modifice variabila x înainte ca threadul f1 sa-şi termine execuţia / afişarea.

1.3

Rezultat

```
Oricare din combinatiile 1 1 ; 1 2 ; 2 1 ; 2 2 The End
Depinde de disponibilitatea procesorului la momentul executiei.
```

Folosind metode syncronizate, alocam ca şi variabilă de execuţie (protejata de accesări străine în momentul execuţiei) chiar adresa de referintă a instanţei clasei ce începe să se execute pe un thread nou. Astfel variabila statică x asociată clasei nu este protejată împotriva accesărilor de către alte threaduri şi execuţia ajunge să se desfăşoare identic cu cea de la punctul 1.2

2.1

Code:

```
#include <iostream>
using namespace std;

class C{
  private:
    int x;
  public:
    C(int i=0){x=i;}
    C& operator++(){ ++x; return *this; }
    C& operator--(){ --x; return *this; }
    int getX() const{ return x; }
};

ostream& operator<<(ostream& stream, const C& obj){
  stream << obj.getX() ;
  return stream;
}

int main(){
  C i;
  cout<<i<<endl;
  cout<<++(++i)<<endl<<i<<endl;
  cout<<--(--i)<<endl<<i<<endl;
}
```

Rezultat:

```
0
2
2
0
0
```

2.2

In main creăm o instanţă a clasei C, denumită i, al cărie valoare este, by default, 0. După primul apel al operatorului "<<" afişăm valoarea lui i, adică 0. La al doilea "cout" apelăm de două ori operatorul "++", care prin definiţie, adaugă valorii lui i cu o unitate per apel. Apelând de doua ori operatorul "<<" rezultă în afişarea de două ori a valorii lui i, care acum este 2, ţinând cont de ordinea de execuţie a operatorilor. Analog pentru ultimul "cout" şi apelul operatorului "--", care, prin definiţie, scade valoarea lui i cu o unitate, iar în final afişăm noua valoare a lui i, care acum este 0 (tot de două ori).

3.

Rezultat

```
Start
B::v()
A::s()
B::w()
B::v()
A::s()
B::w()
A::v()
A::s()
A::w()
```

La execuţia programului creăm două instanţe, una pentru clasei A numită a şi una pentru clasa B, descendent al clasei A, numită b. Iniţializăm instanţa b apelând constructorul default aferent clasei, apoi asociem zonei de memorie a instanţei b şi legătura către instanţa a.

La prima execuţie a->v() este apelata metoda "overloaded" din B, metoda s() moştenită din A şi metoda w() "overloaded" din B. Următorul rând cu `a=(A*)b;` are exact aceeaşi interpretare ca şi la punctul anterior, o variabilă de tipul superclasei i se alocă o zonă de referinţă de tipul subclasei.

La ultima linie `((A)(*b)).v()` se face conversia completă a subclasei la superclasă, şi deci, vor fi apelate metodele cu interpretarea acesteia.

4.

```
public class Main {
  public static void main(String[] args) {
    Interf ma = new Impl();
    try{
      ma.verify(99);
      ma.verify(101);
    }catch(Exception e){
      System.out.println("Valoare incorecta, mai mare decat 100"); }
  }
}
interface Interf{
  public void verify(int i) throws Exception;
}
public class Impl implements Interf{
  @Override
  public void verify(int i) throws Exception {
    if(i>100)
      throw new Exception("Valoare incorecta, mai mare decat 100");
    else
      System.out.println("Valoare corecta, mai mica decat 100");
  }
}
```