Programowanie wizualne

opracował: Paweł Wojciechowski 2023/24

Lab 3

Zagadnienia do opanowania:

- interfejsy
- użycie interfejsów `IEnumerable, IEnumerator, IComparable, Icomparer`
- użycie biblioteki `System.Reflection`

Temat: Interfejsy

1. Pobrać plik Workers.cs albo wykorzystać klasy Worker, Manager i Supervisor z wcześniejszych zajęć. Jeżeli na poziomie klasy Worker metoda Print będzie virtual, dla Manager-> override i dla Supervisor -> new. Która z metod zostanie wywołana dla poniższego przykładu:?

   ```
   Manager m = new Supervisor() { ID = 11 };
   m.Print();
   ```

2. Zdefiniować interfejs `IPrintable` z metodą `Print()` i właściwością ID;
3. Zaimplementować interfejs `IPrintable` tak, żeby działała we wszystkich klasach: `Worker`, `Manager`, `Supervisor.` Metoda powinna wyświetlać informację o klasie z której jest wywołana i ID. Czy trzeba implementować metodę print i właściwość ID we wszystkich wspomnianych klasach żeby kod się kompilował?
4. Zaimplementować interfejs `IPublicPrintable` z metodą `Print`(), która nie wyświetla ID. Zaimplementować interfejs `IPublicPrintable` we wszystkich klasach.
   Sprawdzić co zostanie wyświetlone przy następujących wywołaniach:

   ```
   Supervisor m = new Supervisor() { ID = 11 };
   m.Print();
   ((IPrintable)m).Print();
   ((IPublicPrintable)m).Print();
   ```

   Dlaczego tak jest?
5. Utworzyć klasę `Office`, która w konstruktorze utworzyć 6 pracowników na różnych „poziomach” dane powinny być przechowywane w tablicy. Jakiego typu może być ta tablica?
   W klasie `Supervisor` zmienić modyfikator przy metodzie `Print` z `new` na `override`. Można wykorzystać zakomentowany kod z pliku Workers.cs.
6. Umożliwić przeglądanie pracowników zdefiniowanych w ramach klasy `Office` przez instrukcję `foreach` (interfejs `IEnumerable`, `IEnumerator`)
7. Zdefiniować 2 różne sposoby przeglądania: alfabetycznie oraz według stanowiska alfabetycznie – wykorzystanie enumeratorów nazwanych z wykorzystaniem słowa yield (proszę ręcznie zwracać obiekty) z tablicy.
8. Zaimplementować interfejs `IComparable` w klasie `Worker` porównujący wiek pracownika. Przetestować jego działanie dla metody `Array.Sort` i tablicy z puntku 5.
9. Zaimplementować nową klasę porządkującą obiekty klasy `Worker` w inny sposób niż w punkcie 9 (użycie interfejsu `IComparer`) np. po wieku, a następnie po nazwisku.
10. Zdefiniować interfejs IDraw z jawną implementacją metody Draw (wyświetlającą coś na ekranie).
    - Zaimplementować klasę Obrazek implementującą interfejs IDraw (bez implementacji metody Draw). Spróbować wywołać metodę Draw.
    - Zaimplementować metodę Draw w klasie Obrazek. Która metoda teraz zostanie wywołana? Co jeśli obiekt klasy Obrazek zostanie zrzutowany na interfejs IDraw?
    - Dodać jawną implementację interfejsu IDraw w klasie Obrazek. Czy mogą być dwie metody Draw? (jedna implementująca jawnie interfejs IDraw, a druga zwykła). Która metoda się wywoła i czy rzutowanie coś zmienia?
    - Zaimplementować nową klasę np. DuzyObrazek dziedziczącą po Obrazek i implementującą interfejs IDraw. Która metoda Draw zostanie dla obiektu tej klasy wywołana (jawna implementacja interfejsu, czy metoda klasy nadrzędnej)? Czy rzutowanie na interfejs coś zmienia? Czy obecność jawnej implementacji metody Draw w klasie Obrazek coś zmienia?

Pytania kontrolne:

- Czy w przypadku gdy jeden interfejs dziedziczy po drugim, mogą oba mieć taką samą metodę?
- Jakie metody wymagane są przez interfejs `IEnumerator?`
- Jak w konstrukcji `foreach` wybrać z którego enumeratora chcemy skorzystać?
- Jeżeli interfejs ma jawną implementację metody to czy można wywołać tę metodę dla obiektu (bez rzutowania na interfejs)?

11. Wykorzystując bibliotekę System.Reflection, wyświetlić wszystkie typy zdefiniowane w bibliotece HiddenLibrary.dll. Biblioteki nie należy dodawać do projektu ani oglądać w programach typu ildasm.exe.
12. Sprawdzić, która klasa implementuje jedyny interfejs zdefiniowany w bibliotece. Utworzyć obiekt tej klasy. Wyświetlić wszystkie metody tej klasy i wywołać metodę na literę S.