```
public class Dwarf
{
    public TypeEnum Type { get; private set; }
    public BankAccount BankAccount;
    public Backpack Backpack;
    IBehaviorOnTheShaft shaftStrategy;

    public Dwarf(TypeEnum type)
    {
        Type = new TypeEnum();
        Backpack = new Backpack();
        BankAccount = new BankAccount();
        Type = type;
        if (type == TypeEnum.Saboteur) shaftStrategy = new SaboteurStrategy();
        else shaftStrategy = new CommonDwarfStrategy();
    }

    public void Working(Shaft shaft)
    {
        shaftStrategy.DoYourJob(shaft, this.Backpack);
    }
}
```

Wykorzystanie strategii w klasie Dwarf, umożliwia w klasie Mine, użycia tylko jednej metody "Working" z klasy Dwarf, która ma różne zachowanie w zależności od typu krasnala wybranego przy tworzeniu instancji klasy Dwarf. Dzięki temu rozwiązaniu nie trzeba tworzyć kilku różnych metod w klasie Dwarf o różnych zachowaniach, co sprawia, że kod jest bardziej hermetyczny i posiada zasadę pojedynczej odpowiedzialności.

Fragment kodu z klasy Mine

```
private void DwarfsDoJob(Shaft shaft)
{
    City.newsPaper.Add("Mine: " + shaft.dwarfs.Count + " dwarfs go to the shaft.");
    foreach (var dwarf in shaft.dwarfs)
    {
        dwarf.Working(shaft);
    }
}
```

Testy sprawdzające działanie strategii

```
[TestMethod]
public void ShaftShouldBeDestroyedWhenOnTheShaftIsSaboteur()
{
    //Given
    Dwarf saboteur = new Dwarf(TypeEnum.Saboteur);
    Shaft shaft = new Shaft();

    //When
    saboteur.Working(shaft);

    //Then
    Assert.AreEqual(shaft.isExist, false);
}
[TestMethod]
public void WhenCommonDwarfWorkingShaftShouldntBeDestroyed()
{
    //Given
    Dwarf common = new Dwarf(TypeEnum.Father);
    Shaft shaft = new Shaft();

    //When
    common.Working(shaft);

    //Then
    Assert.AreEqual(shaft.isExist, true);
}
```