Je dán následující program:

```
class ClassA {
  virtual public String WhoAreYou() {
    return("I am ClassA");
  }

public static void Main(String[] args) {
    ClassA a = new ClassA();
    PrintWhoIsIt(a);
    ClassB b = new ClassB();
    PrintWhoIsIt(b);
  }

static void PrintWhoIsIt(ClassA p) {
    Console.WriteLine(p.WhoAreYou());
  }
}
```

Zapište kompletní kód třídy ClassB tak, aby výsledkem volání uvedené metody Main byl následující výpis:

```
I am ClassA
I am ClassB
```

Program nesmí skončit předčasně!

Mějme následující program:

```
class MyClassA {
  virtual public void PrintSomething() {
    Console.WriteLine("42");
  }
}

class MyClassB : MyClassA{
  override public void PrintSomething() {
    Console.WriteLine("24");
  }
}

class MyClassC : MyClassB{
  override public void PrintSomething() {
    Console.WriteLine("0");
  }
}
```

Co vypíše následující úsek programu?

```
MyClassA mc1 = new MyClassA();
MyClassA mc2 = new MyClassB();
MyClassA mc3 = new MyClassC();
MyClassB mc4 = (MyClassB) mc2;
mc1.PrintSomething();
mc2.PrintSomething();
mc3.PrintSomething();
mc4.PrintSomething();
Console.WriteLine(mc1 is MyClassA);
Console.WriteLine(mc1 is MyClassB);
Console.WriteLine(mc2 is MyClassA);
Console.WriteLine(mc2 is MyClassB);
Console.WriteLine(mc3 is MyClassA);
Console.WriteLine(mc3 is MyClassB);
Console.WriteLine(mc4 is MyClassA);
Console.WriteLine(mc4 is MyClassB);
```

Mějme následující program:

```
abstract class MyClassA {
  void PrintSomething() {
    Console.WriteLine("42");
  }
  abstract public void PrintSomethingElse();
}

class MyClassB : MyClassA{
  public void PrintSomethingElse() {
    Console.WriteLine("24");
  }
}
```

U každé z následujících řádek určete, zda je v pořádku, popř. zda způsobí chybu při překladu nebo chybu za běhu programu. Při uvažování každé další řádky předpokládejte, že bezchybné předchozí řádky byly vykonány, zatímco chybové řádky v programu vůbec nebyly.

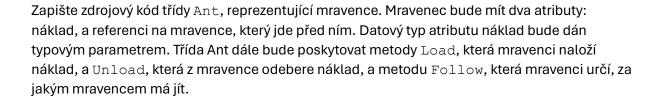
	ОК	Chyba při překladu	Chyba za běhu
<pre>MyClassA mca1 = new MyClassA();</pre>			
<pre>MyClassA mca2 = new MyClassB();</pre>			
<pre>MyClassB mcb1 = new MyClassB();</pre>			
MyClassB mcb2 = (MyClassB)mca2;			
<pre>mca2.printSomething();</pre>			
<pre>mca2.printSomethingElse();</pre>			
<pre>mcb2.printSomething();</pre>			
<pre>mcb2.printSomethingElse();</pre>			

(v každé řádce tabulky zapište jeden křížek do sloupce odpovídajícího situaci, která nastane)

Je dána třída Bag, reprezentující neuspořádanou množinu prvků, implementující následující rozhraní:

```
interface IBag {
  void Add(Object o);
  void Remove(Object o);
  Object RandomItem();
}
```

Kromě metod rozhraní poskytuje třída Bag ještě bezparametrický konstruktor. Zapište obalovací třídu HomogeneousBag s typovým parametrem, která poskytne všechny metody rozhraní, ale umožní pouze práci s datovým typem určeným typovým parametrem. Zapište celou implementaci třídy HomogeneousBag.



Zapište zdrojový kód třídy Caravan, reprezentující karavanu mravenců. Datový typ pro náklad bude specifikován typovým parametrem třídy Caravan. Třída poskytne metody Attach a Detach, které připojí/odpojí jednoho mravence z karavany. Metody přebírají/vracejí datovou položku odpovídající typu nákladu mravence.