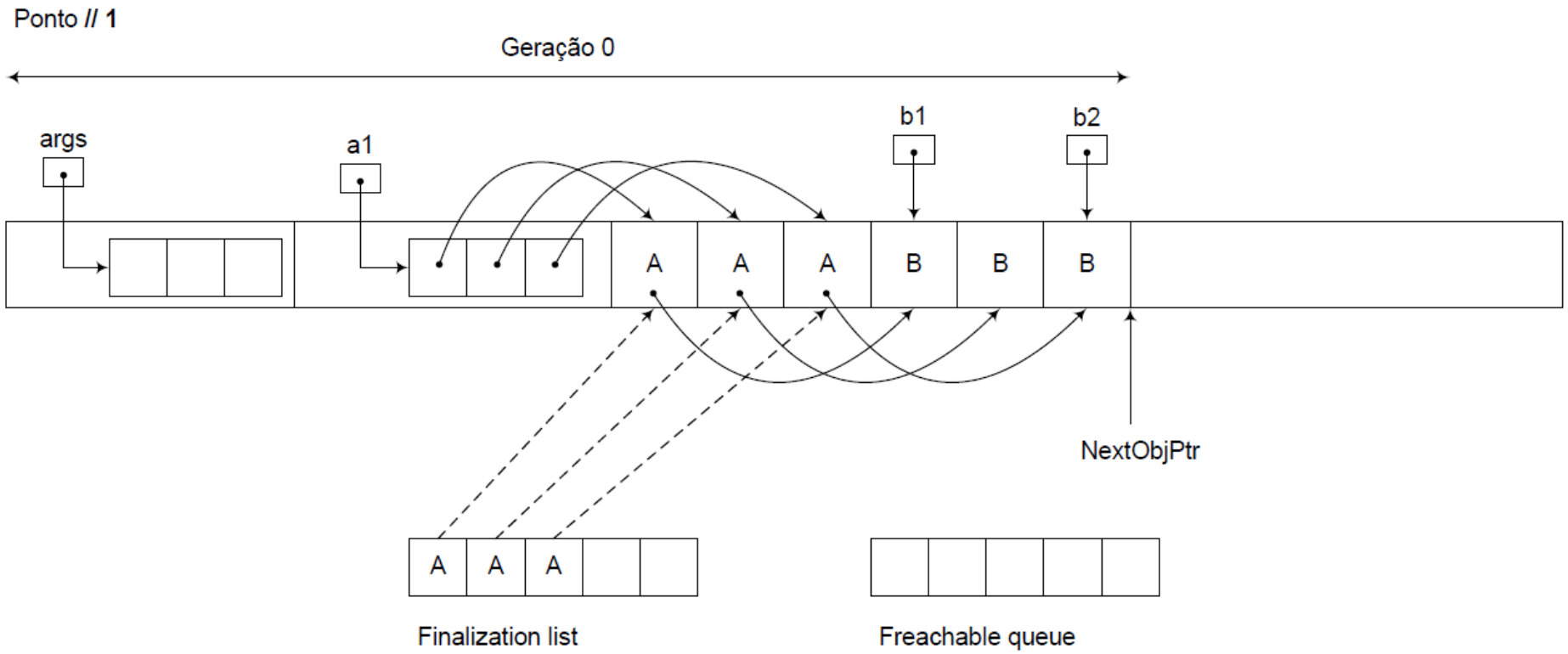
Exercício 1

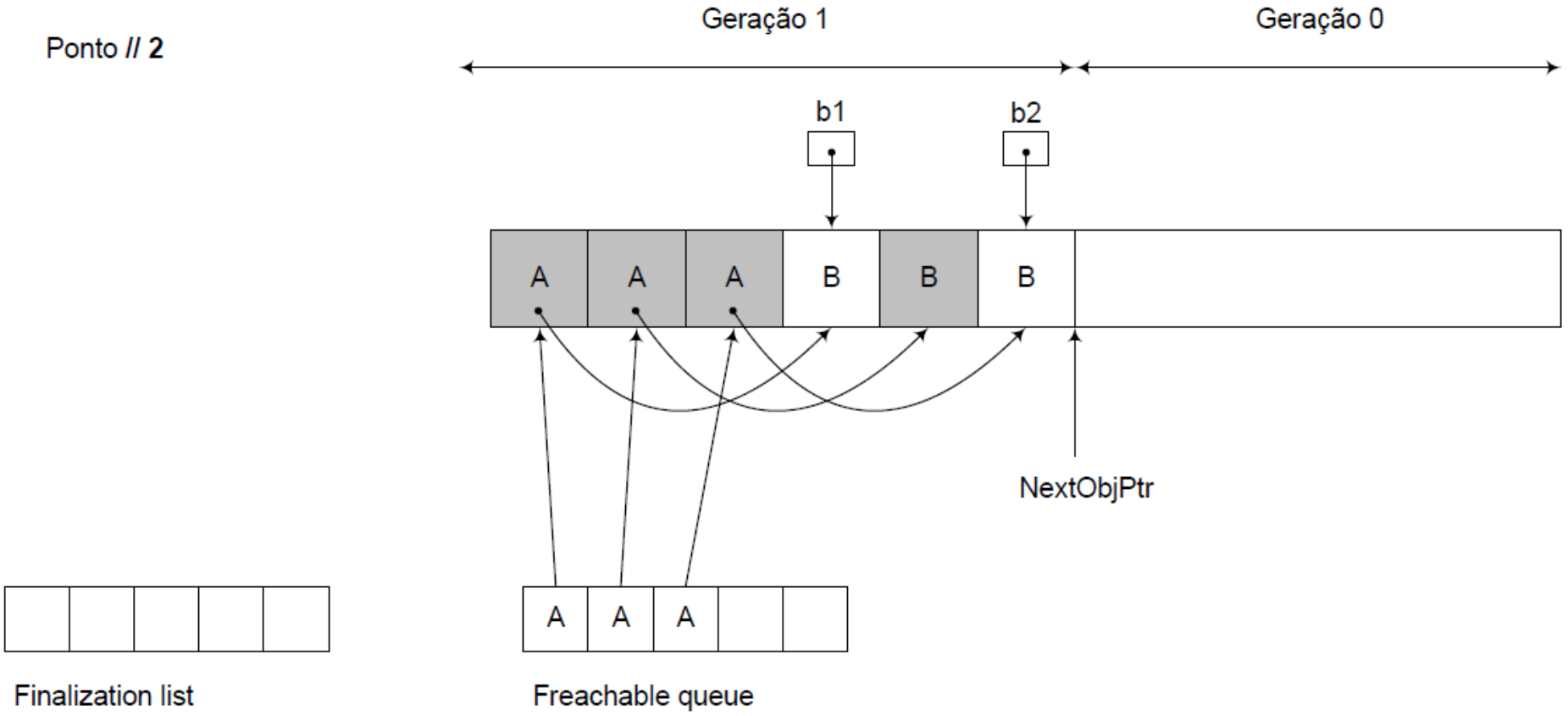
Analise o seguinte código. Apresente na forma que considere conveniente o estado do *heap* (relativo aos objectos criados durante a execução do método Main e organizado por gerações) nos pontos identificados pelos comentários.

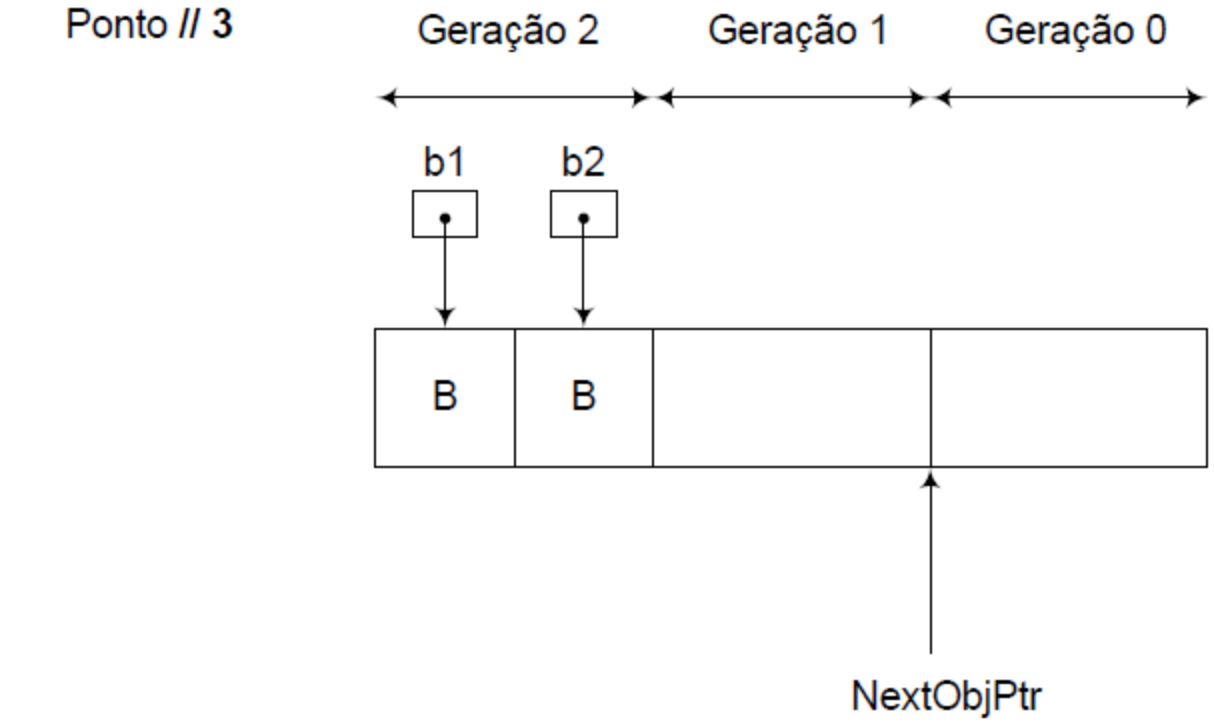
```
class A {
B b;
public A() { b = new B(); }

public B myB { get { return b; } }

~A() {}
}

class B {}
```

```
class Program{
static void Main(string[] args) {
 A[] a1 =new A[3];
 for (int i = 0; i < a1.Length; ++i) a1[i] = new A();
 B b1 = a1[0].myB, b2 = a1[a1.Length-1].myB;
//1
GC.Collect();
//2
GC.WaitForPendingFinalizers();
GC.Collect();
//3
Console.WriteLine(b1.GetHashCode());
Console.WriteLine(b2.GetHashCode());
}
}
```


Ponto // 1
Geração 0
args
a1
b1
b2
A
A
A
B
B
B
NextObjPtr
A
A
A
Finalization list
Freachable queue

Ponto // **2**


Geração 1
Geração 0
b1
b2
A
A
A
B
B
B
NextObjPtr
A
A
A
Finalization list
Freachable queue


Ponto // 3
Geração 2
Geração 1
Geração 0
b1
b2
B
B
NextObjPtr
Finalization list
Freachable queue

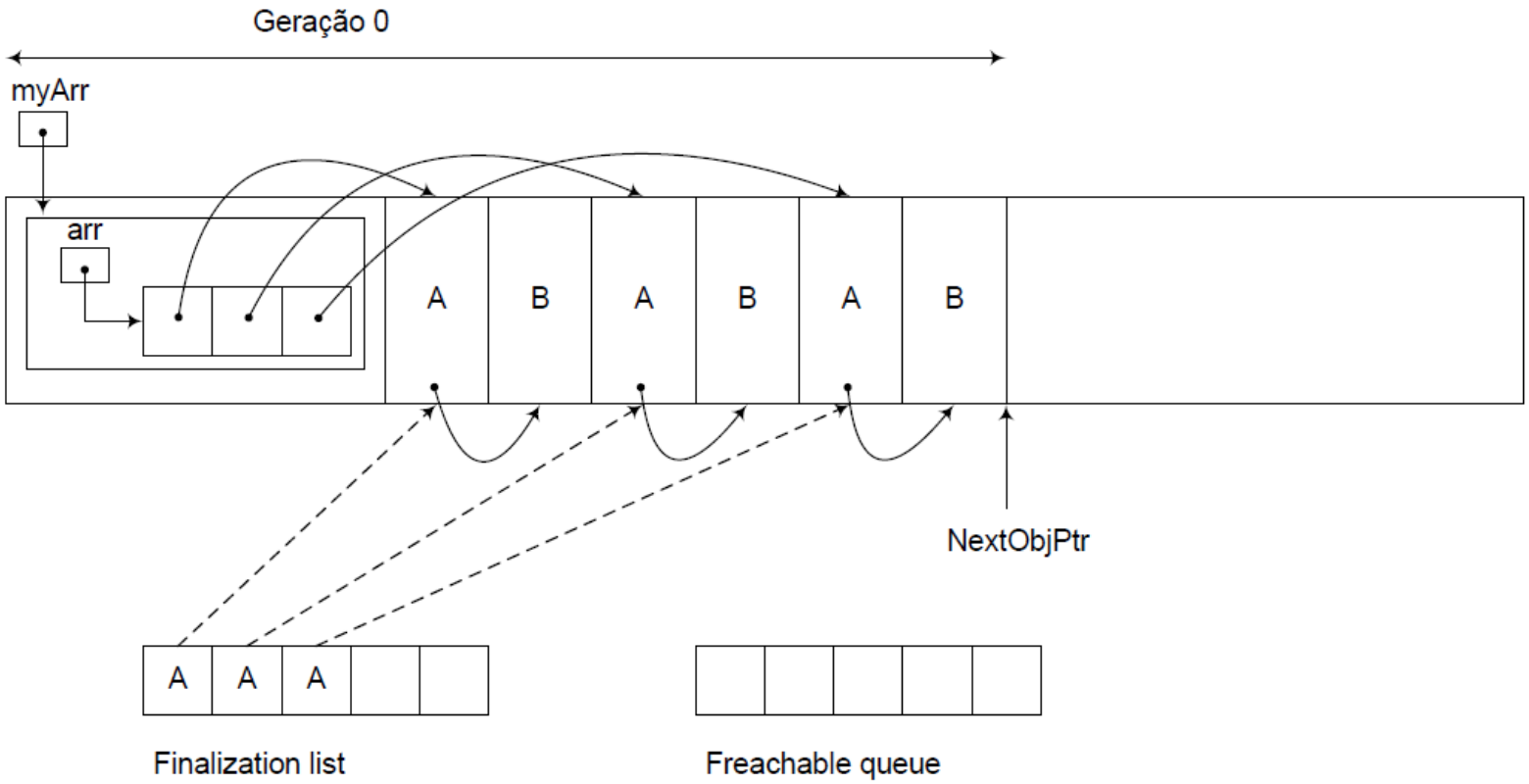
Exercício 2

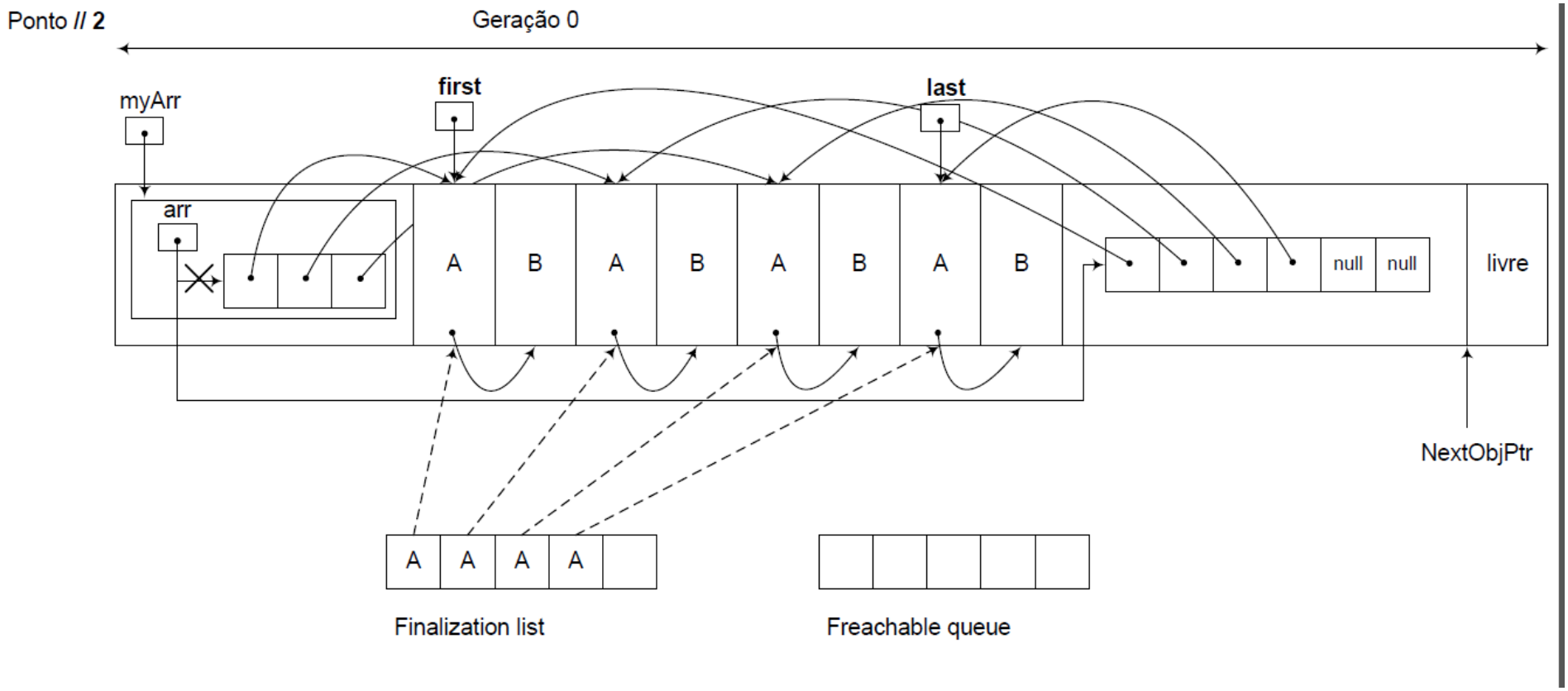
Analise o seguinte código. Apresente na forma que considere conveniente o estado do heap (relativo aos objectos criados durante a execução do método Main e organizado *por gerações) nos pontos identificados pelos comentários. Assuma que o programa foi compilado em modo release*.

```
class MyArray {
   private A[] arr;
   private int size;
   public MyArray(int dim) {
      arr = new A[dim]; size = 0;
    }
  private void Grow() {
     A[] newArr = new A[size*2];
    Array.Copy(arr, newArr, size);
     arr = newArr;}
  public void Add(A a) {
     if (size == arr.Length) Grow();
     arr[size++] = a;
}
public A Get(int i) { return arr[i]; }
}
class B { }
```

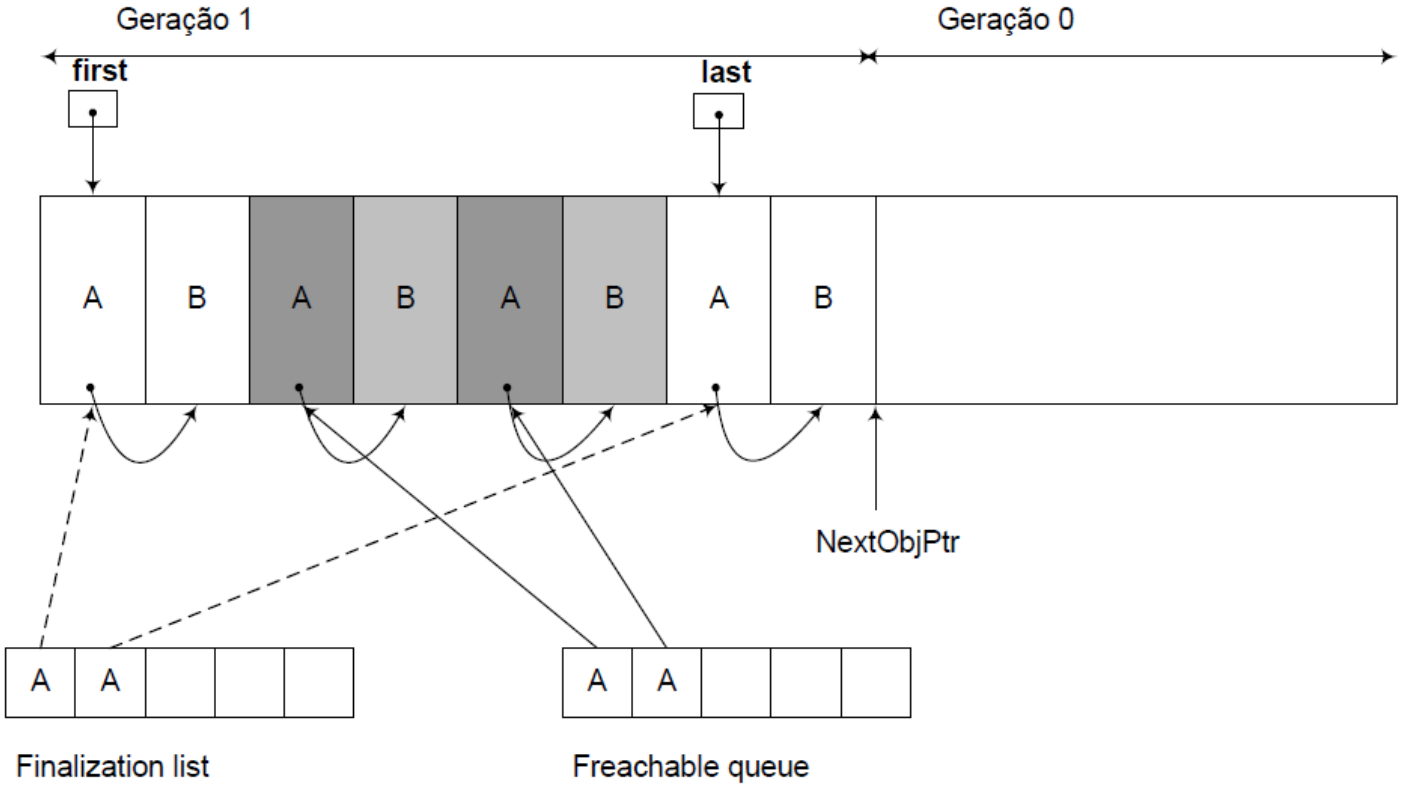
```
class A {
    B b;
    public A() { b = new B(); }
     ~A() { }
}
class Program {
    static void Main() {
        MyArray myarr = new MyArray(3);
        for (int i = 0; i < 3; ++i) myarr.Add(new A());
         //1
          myarr.Add(new A());
           A first=myarr.Get(0), last=myarr.Get(3);
        //2
        GC.Collect();
       //3
        GC.WaitForPendingFinalizers();
         GC.Collect();
        //4
Console.WriteLine(first.ToString());
Console.WriteLine(last.ToString());}}
```

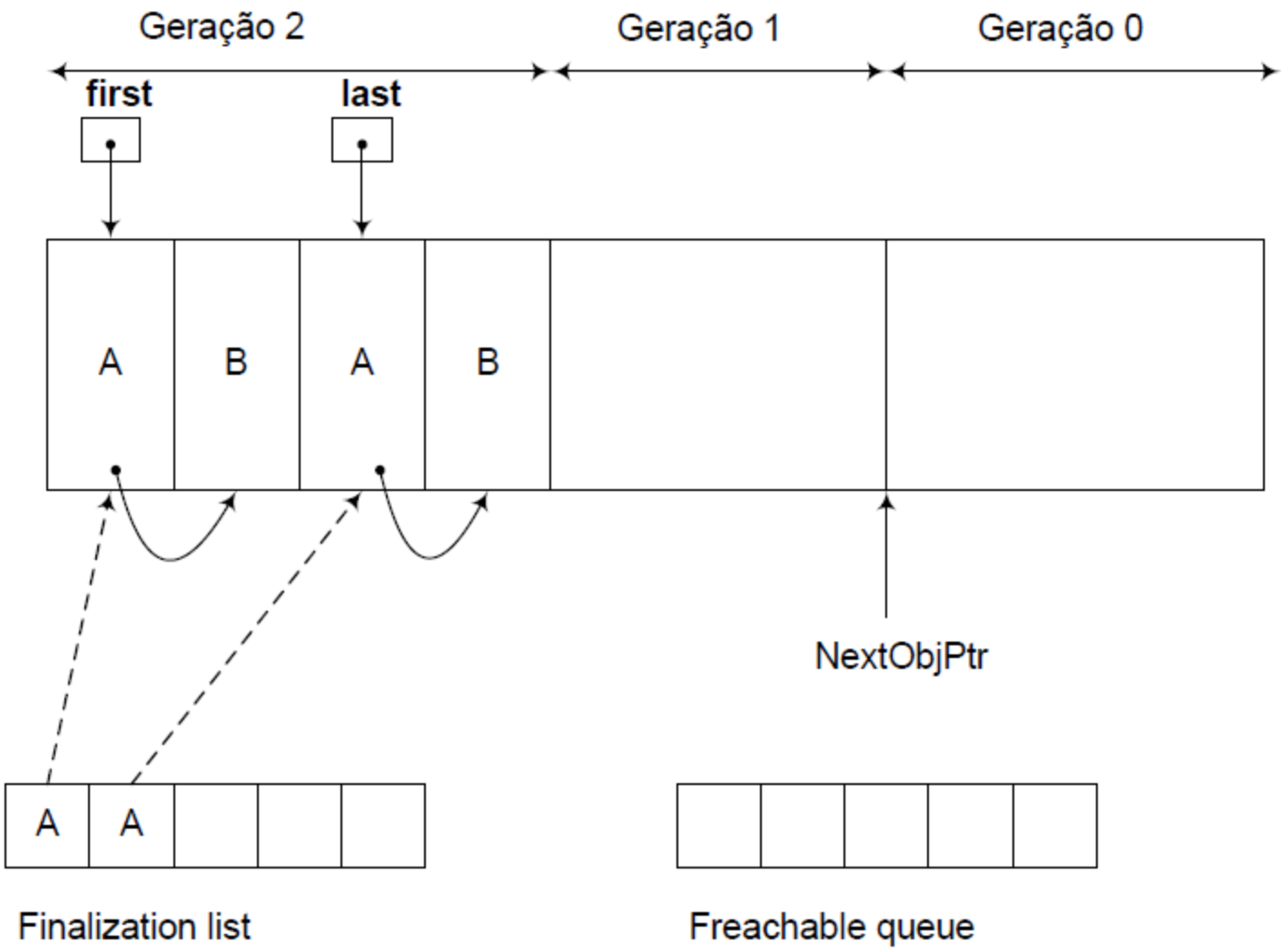
Ponto // **1**


Geração 0
myArr
arr
A
B
A
B
A
B
NextObjPtr
A
A
A
Finalization list
Freachable queue


Ponto // 2
Geração 0
myArr
first
last
arr
A
B
A
B
A
B
A
B
null
null
livre
NextObjPtr
A
A
A
A
Finalization list
Freachable queue

Ponto // 3


Geração 1
Geração 0
first
last
A
B
A
B
A
B
A
B
NextObjPtr
A
A
A
A
Finalization list
Freachable queue

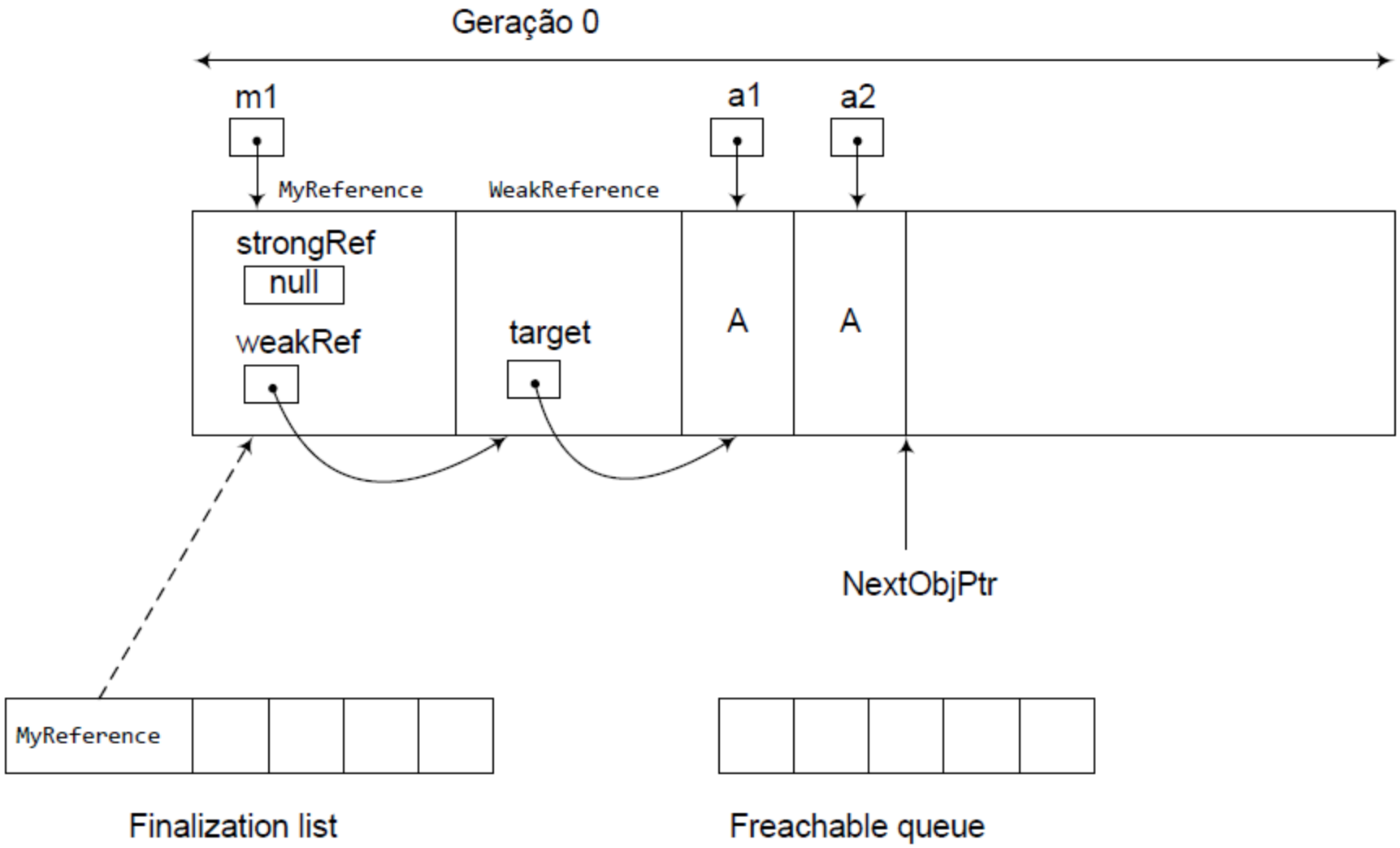

Geração 2
Geração 1
Geração 0
first
last
A
B
A
B
NextObjPtr
A
A
Finalization list
Freachable queue

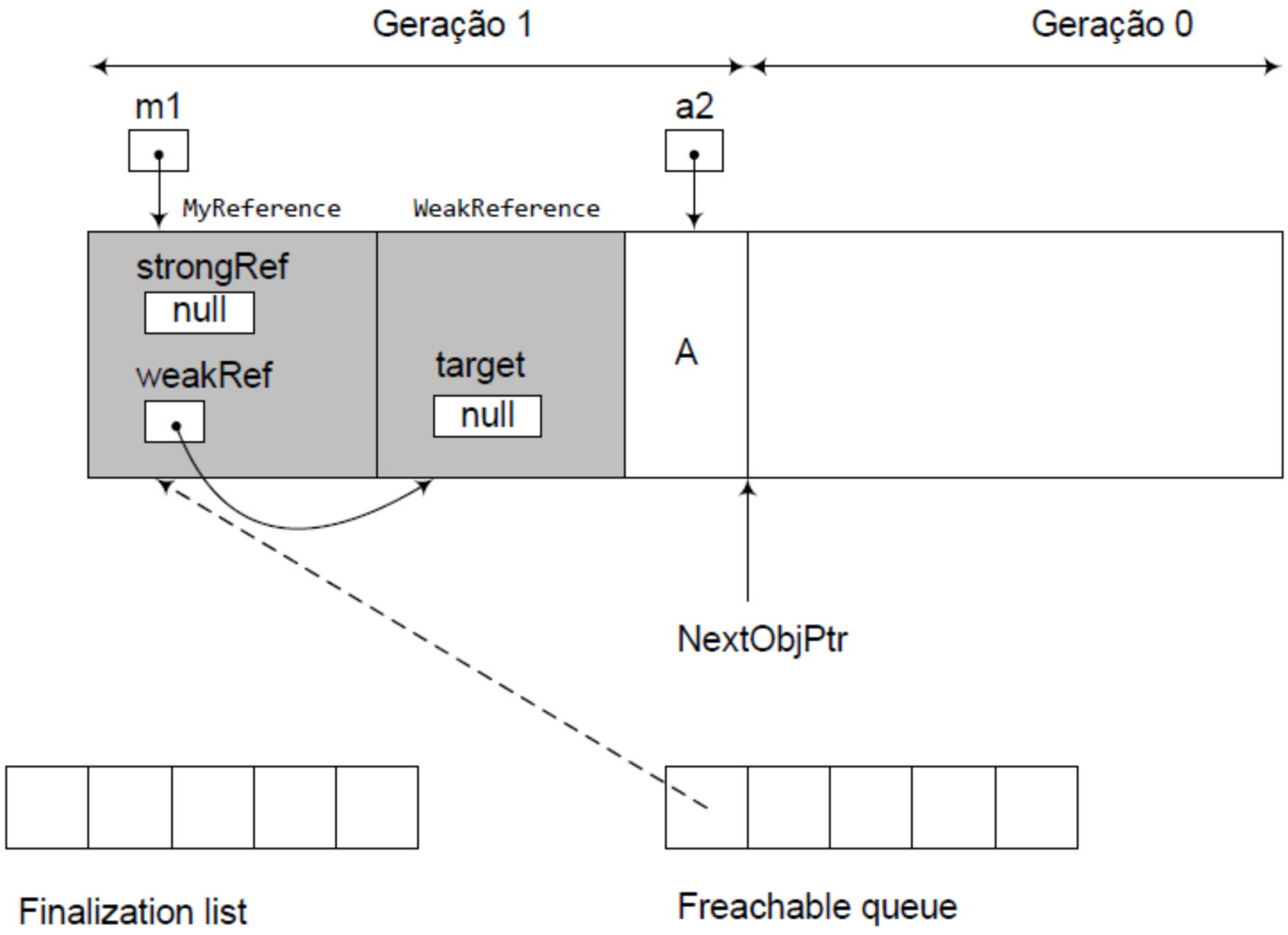
- Exercício 3

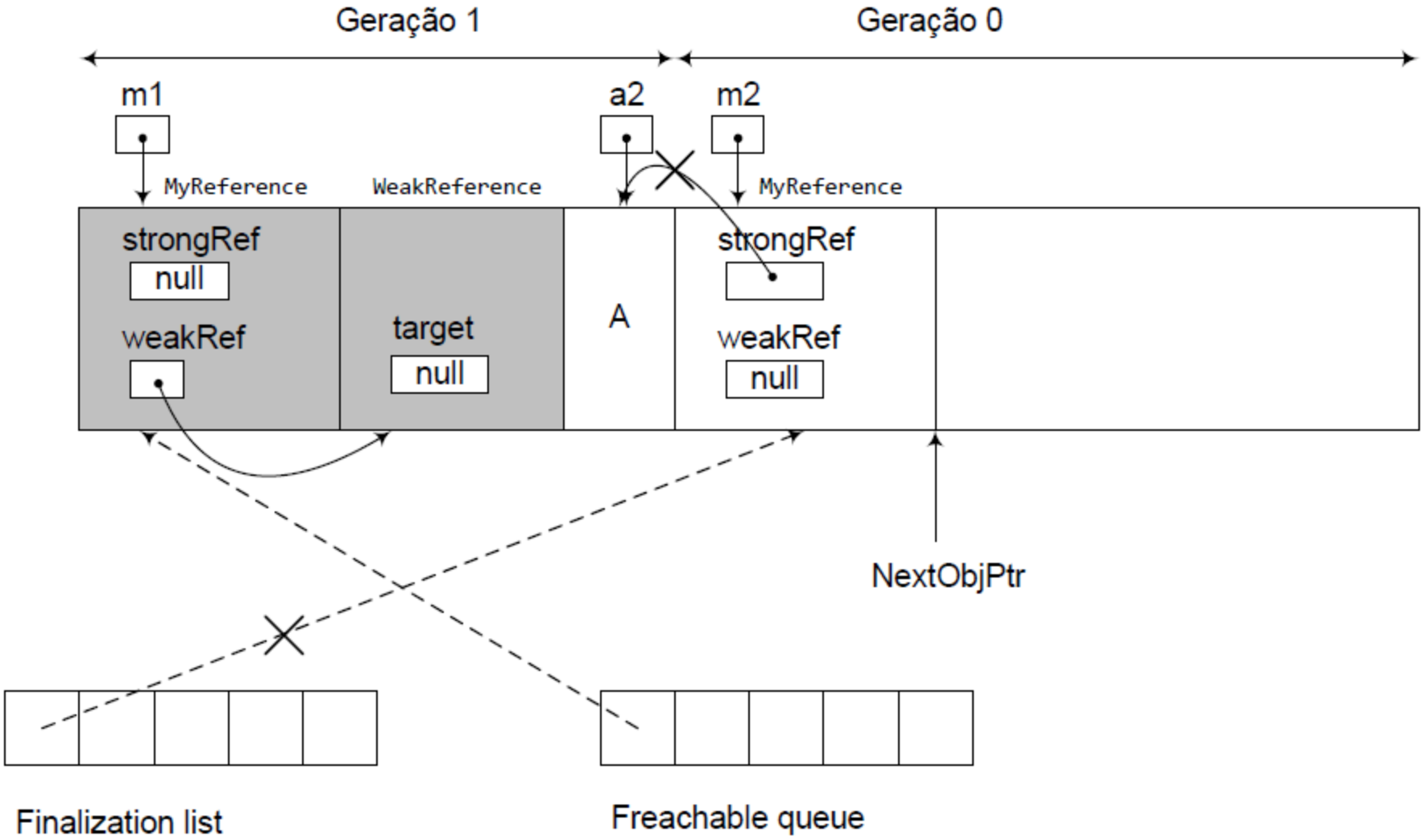
8. [4] Analise o seguinte código. Apresente na forma que considere conveniente o estado do *heap* (raízes, *heaps* das gerações 0, 1 e 2, e *Finalization* e *FReachable queues*) relativo aos objectos criados durante a execução do método `Main` nos pontos identificados pelos comentários. Considere que o código apresentado foi compilado em modo *release* e que não ocorrem ciclos de recolha para além das chamadas ao método `Collect` da classe `GC`.

```
class MyReference  : IDisposable {
   object strongRef;
   WeakReference weakRef;

   public MyReference(object o,
                      bool isStrong) {
      if (isStrong) strongRef = o;
      else weakRef = new WeakReference(o);
   }

   private void Dispose(bool disposing) {
      if (strongRef != null) strongRef = null;
      else weakRef.Target = null;
   }

   public void Dispose() {
      Dispose(true);
      GC.SuppressFinalize(this);
   }

   public object Target {
      get {
        if (strongRef != null) return strongRef;
        return  weakRef.Target;
      }
   }

   ~MyReference() {  Dispose(false); }
 }
```

```
class A {
   public readonly int v;
   public A(int v) { this.v=v; }
   public override string ToString() {
      return v.ToString();
   }
}

class Program {
  static void Main() {
    A a1 = new A(3), a2 = new A(4);

    MyReference m1 = new MyReference(a1, false);
    Console.WriteLine(m1.Target);
    //1
    GC.Collect();
    //2
    using (MyReference m2 = new MyReference(a2,true))
    {
      Console.WriteLine(m2.Target);
    }
    //3
    GC.WaitForPendingFinalizers();
    GC.Collect();
    //4
  }
}
```


Geração 0
m1
a1
a2
MyReference
WeakReference
strongRef
null
weakRef
target
A
A
NextObjPtr
MyReference
Finalization list
Freachable queue


Geração 1
Geração 0
m1
a2
MyReference
WeakReference
strongRef
null
weakRef
target
null
A
NextObjPtr
Finalization list
Freachable queue


Geração 1
Geração 0
m1
a2
m2
MyReference
WeakReference
MyReference
strongRef
null
weakRef
target
null
A
strongRef
weakRef
null
NextObjPtr
Finalization list
Freachable queue