Объектно-ориентированное программирование с использованием Java и C#

Обобщенные типы - generics (Java)

```
public class Stack<T>
{
  private T[] items;
  private int count;

  public Stack(int capacity)
  {
    items = (T[])new Object[capacity];
  }

  public void push(T x)
  {
    items[count++] = x;
  }

  public T pop()
  {
    return items[--count];
  }
}
```

Обобщенные типы - generics (C#)

```
public class Stack<T>
{
  T[] items;
  int count;

  public Stack(int capacity)
  {
    items = new T[capacity];
  }

  public void Push(T x)
  {
    items[count++] = x;
  }

  public T Pop()
  {
    return items[--count];
  }
}
```

Цикл foreach для коллекций

Java:

```
Stack<String> stack = new Stack<String>();
for (String s : stack)
  System.out.println(s);
```

Как это работает внутри:

```
Iterator<String> it = stack.iterator();
while (it.hasNext()) {
  String s = it.next();
  System.out.println(s);
}
```

C#:

```
Stack<string> stack = new Stack<string>();
foreach (string x in stack)
  Console.WriteLine(x);
```

Обобщенные типы (generics)

Java:

```
items = (T[])new Object[size];
```

C#:

```
items = new T[size];
```

В C# можно использовать value-типы как параметры обобщенных типов!

```
var stack = new Stack<int>();
```

Обнуляем (в методе Pop) с помощью `default`:

```
//items[count] = null;
items[count] = default(T);
```

Реализация итератора (Java)

```
public class Stack<T> implements Iterable<T> {

    private T[] items;
    private int count;

    @Override
    public Iterator<T> iterator() { return new StackIterator(count); }

    class StackIterator implements Iterator<T>
    {
        private int pos;

        public StackIterator(int pos) { this.pos = pos; }

        @Override
        public boolean hasNext() { return pos > 0; }

        @Override
        public T next() { return items[--pos]; }
    }
}
```

Реализация итератора (C#)

```
public class Stack<T> : IEnumerable<T>
{
  int count;
  T[] items;

  public IEnumerator<T> GetEnumerator()
  {
    for (int i = count - 1; i >= 0; i--)
      yield return items[i];
  }

  System.Collections.IEnumerator
       System.Collections.IEnumerable.GetEnumerator()
  {
    return GetEnumerator();
  }
}
```

Стандартные коллекции

```
var list = new List<string>();
list.Add("aaa");
list.Add("bbb");
list.Add("ccc");
for (int i = 0; i < list.Count; i++)
  Console.WriteLine(list[i]);

list[2] = "c";
foreach (string s in list)
  Console.WriteLine(s);
```

```
List<String> list = new ArrayList<String>();
list.add("aaa");
list.add("bbb");
list.add("ccc");
for (int i = 0; i < list.size(); i++)
  System.out.println(list.get(i));

list.set(2, "c");
for (String s : list)
  System.out.println(s);
```