UML для программистов

UML (Unified Modeling Language)

- Концептуальный уровень
- Уровень спецификаций
- Уровень реализаций

Концептуальная UML-диаграмма

```
public class Animal { }
public class Dog : Animal { }
```

```
public class TreeMap
    {
        private TreeMapNode topNode = null;

        public void Add(IComparable key, object value)
        {
            if (topNode == null)
                topNode = new TreeMapNode(key, value);
            else
                topNode.Add(key, value);
        }
         public object Get(IComparable key)
        {
            return topNode == null ? null : topNode.Find(key);
        }
    }
```

```
internal class TreeMapNode
    {
        private static readonly int LESS = 0;
        private static readonly int GREATER = 1;
        private IComparable key;
        private object value;
        private TreeMapNode[] nodes = new TreeMapNode[2];
        public TreeMapNode(IComparable key, object value)
        {
            this.key = key;
            this.value = value;
        }
        public object Find(IComparable key)
        {
            if (key.CompareTo(this.key) == 0) return value;
            return FindSubNodeForKey(SelectSubNode(key), key);
        }
```

```
private int SelectSubNode(IComparable key)
        {
            return (key.CompareTo(this.key) < 0) ? LESS : GREATER;
        }

        private object FindSubNodeForKey(int node, IComparable key)
        {
            return nodes[node] == null ? null : nodes[node].Find(key);
        }

        public void Add(IComparable key, object value)
        {
            if (key.CompareTo(this.key) == 0)
                this.value = value;
            else
                AddSubNode(SelectSubNode(key), key, value);
        }
```

```
private void AddSubNode(int node, IComparable key, object value)
        {
            if (nodes[node] == null)
                nodes[node] = new TreeMapNode(key, value);
            else
                nodes[node].Add(key, value);
        }
```

Диаграммы классов

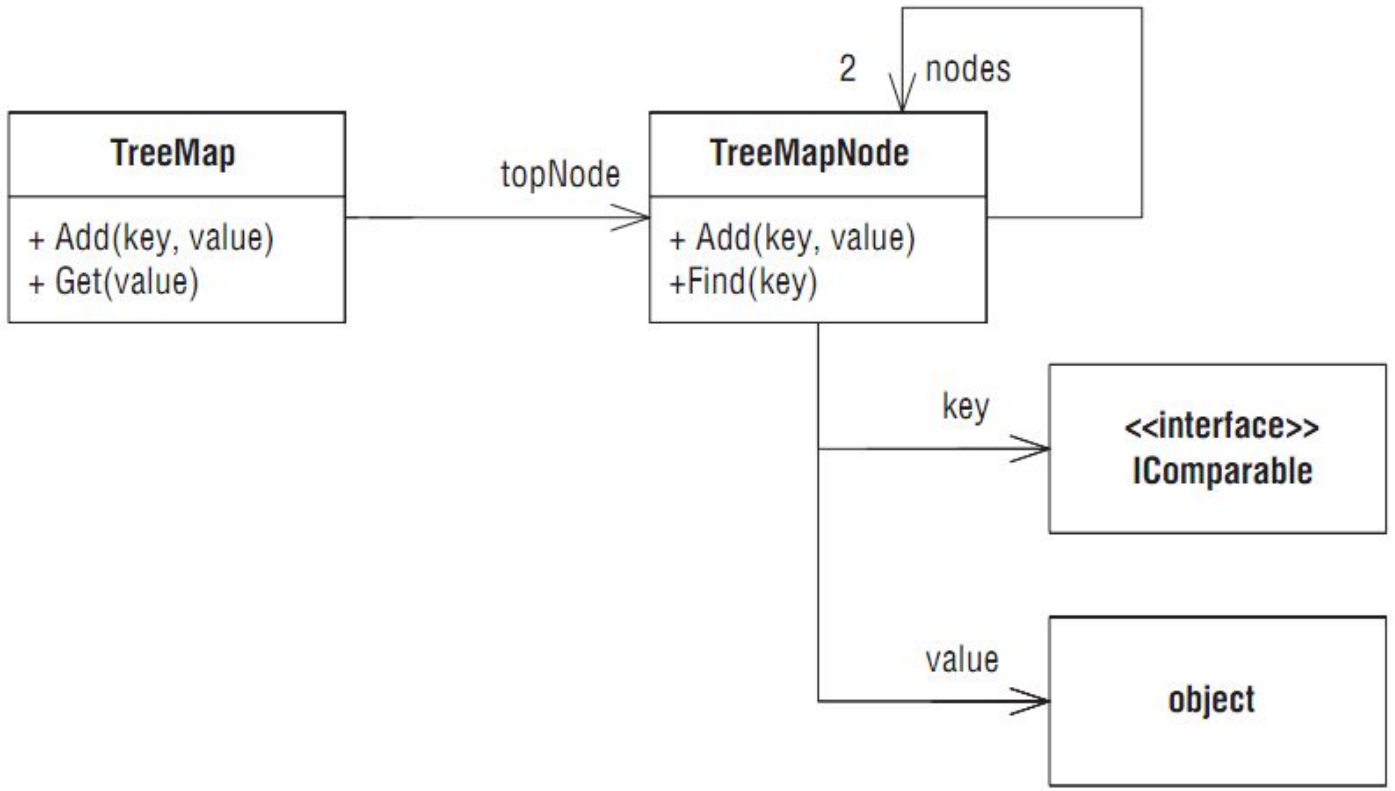


Диаграмма классов для TreeMap

- Классы представляются прямоугольниками, а отношения между ними – стрелками.
- На данной диаграмме все отношения являются ассоциациями.
- Имя над ассоциацией соответствует имени переменной, в которой хранится ссылка.
- Число рядом с острием стрелки обычно говорит о том, сколько экземпляров может быть связано данным отношением. Если это число больше 1, то, как правило, имеется в виду массив.
- В прямоугольниках классов может быть несколько отделений. В верхнем отделении всегда записывается имя класса, а в остальных содержатся функции и переменные.
- Нотация «interface» означает, что IComparable – интерфейс.
- Большая часть остальных показанных обозначений необязательна.

Диаграммы объектов

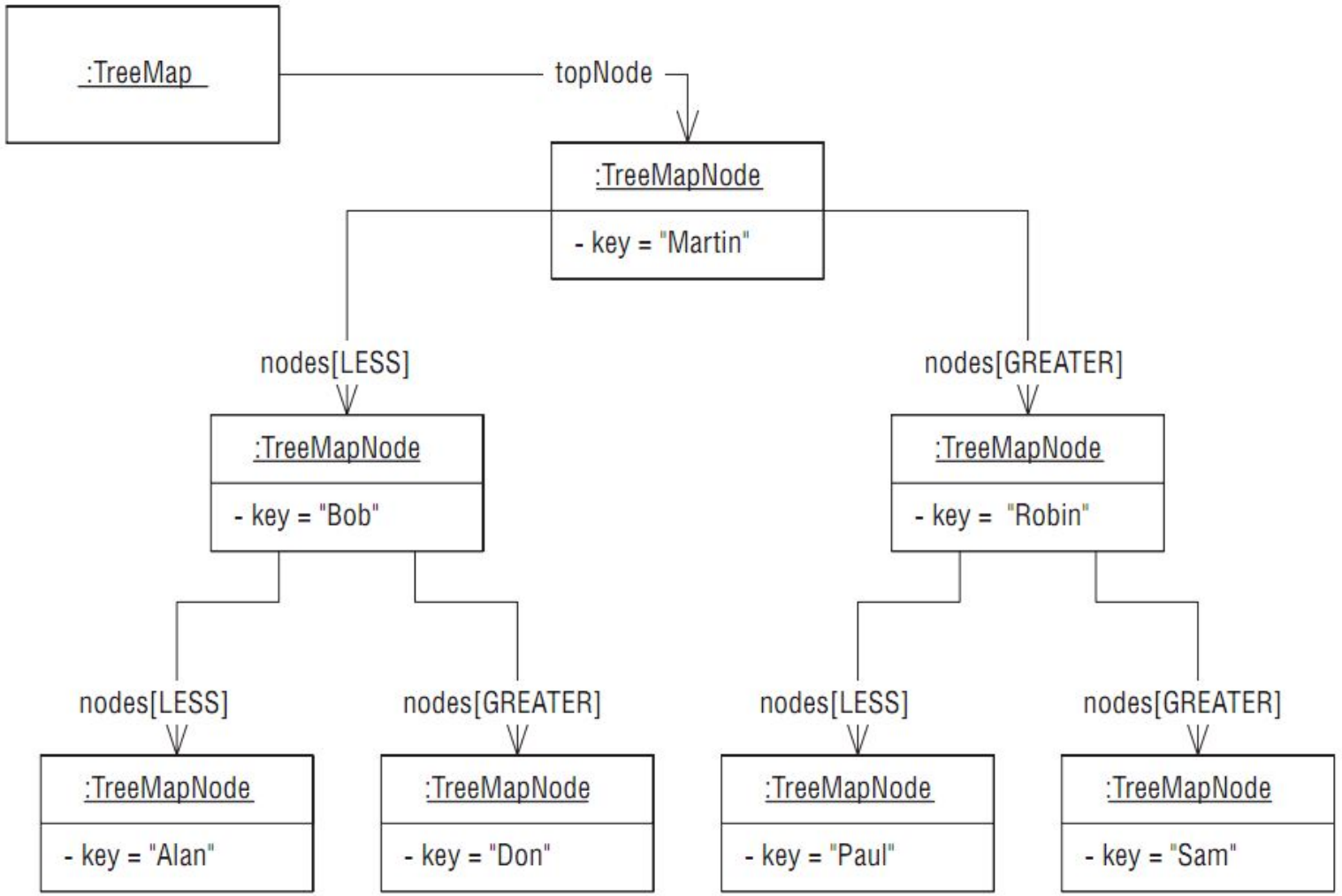
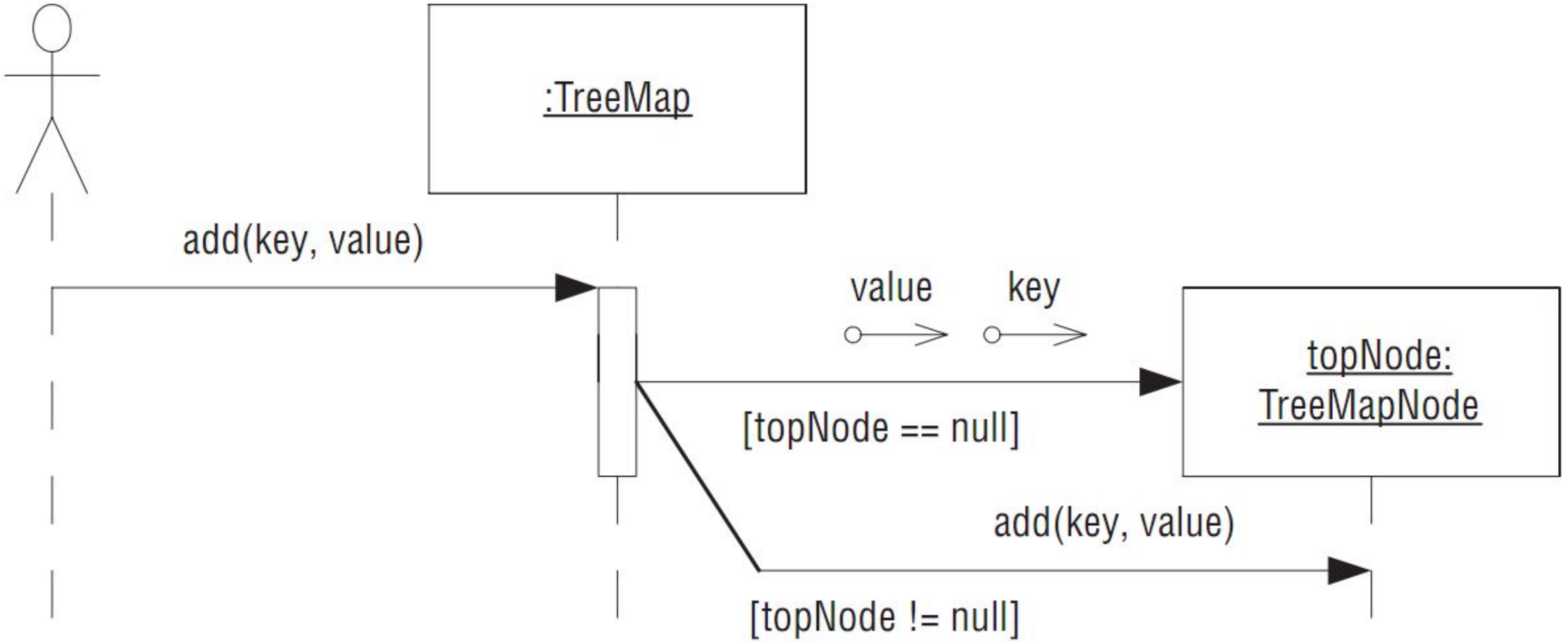


Диаграмма объектов для TreeMap

Диаграммы последовательности



Метод TreeMap.Add

Диаграммы кооперации

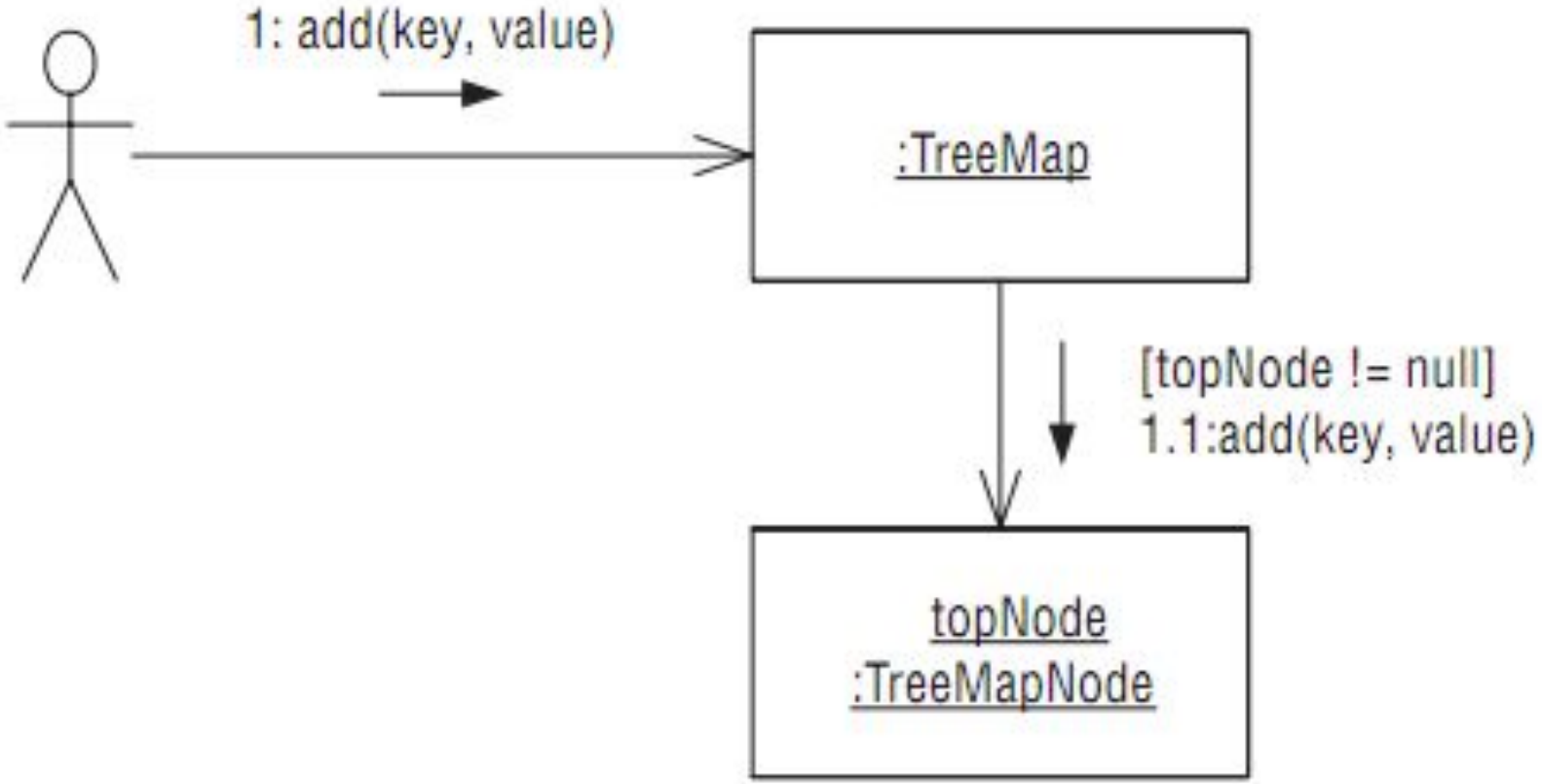
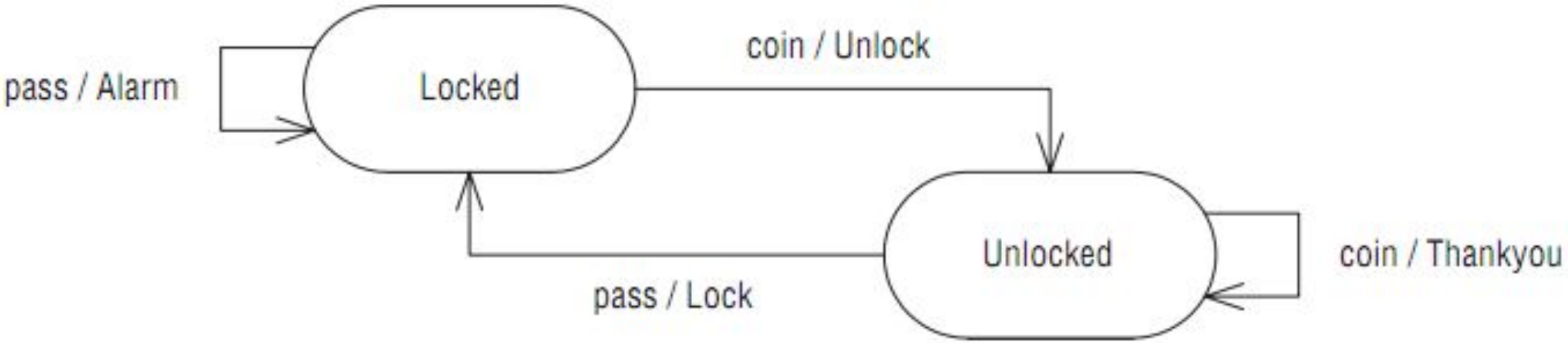


Диаграмма кооперации в одном из случаев выполнения метода TreeMap.Add

Диаграммы состояний

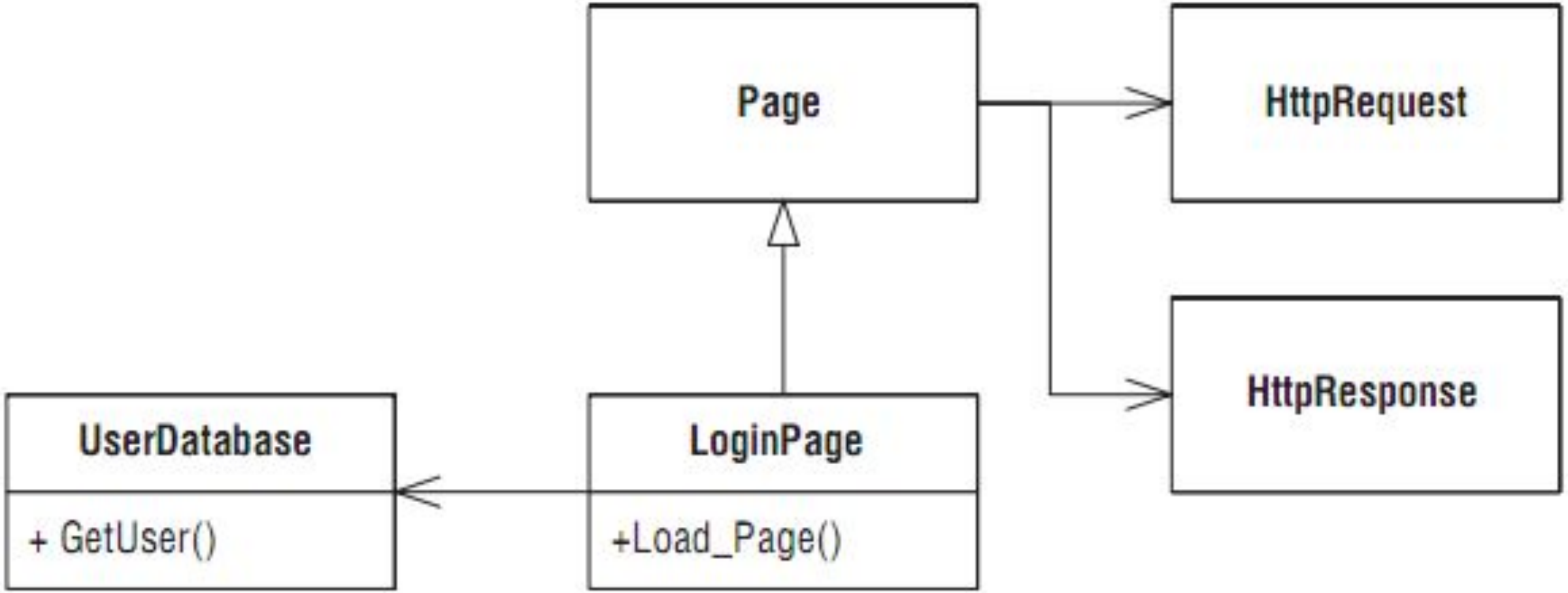


Конечный автомат, описывающий работу турникета в метро

Работа с диаграммами

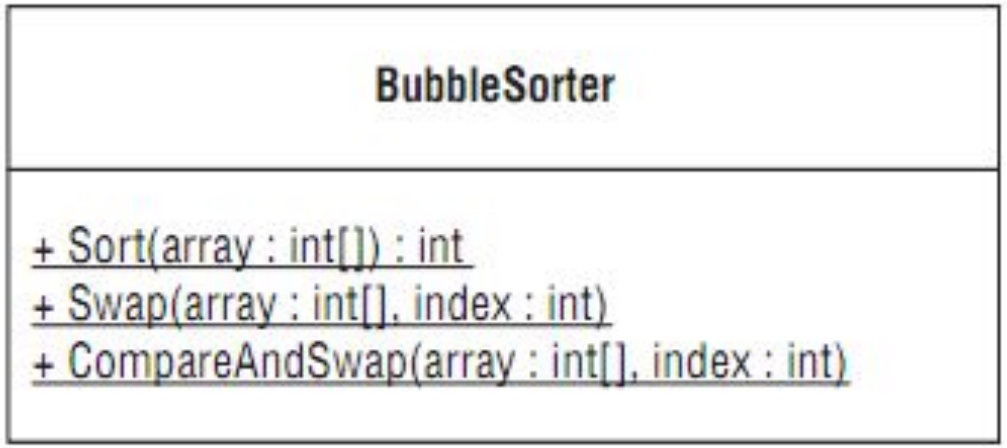
- Зачем нужно моделирование
- Зачем строить модели программ

Общение с другими людьми



Класс LoginPage

```
public class BubbleSorter
{
    private static int operations;
    public static int Sort(int[] array)
    {
        operations = 0;
        if (array.Length <= 1)
            return operations;
        for (int nextToLast = array.Length - 2;
          nextToLast >= 0; nextToLast--)
            for (int index = 0; index <= nextToLast; index++)
                CompareAndSwap(array, index);
        return operations;
    }
    private static void Swap(int[] array, int index)
    {
        int temp = array[index];
        array[index] = array[index + 1];
        array[index + 1] = temp;
    }
    private static void CompareAndSwap(int[] array, int index)
    {
        if (array[index] > array[index + 1])
            Swap(array, index);
        operations++;
    }
}
```



Класс BubbleSorter

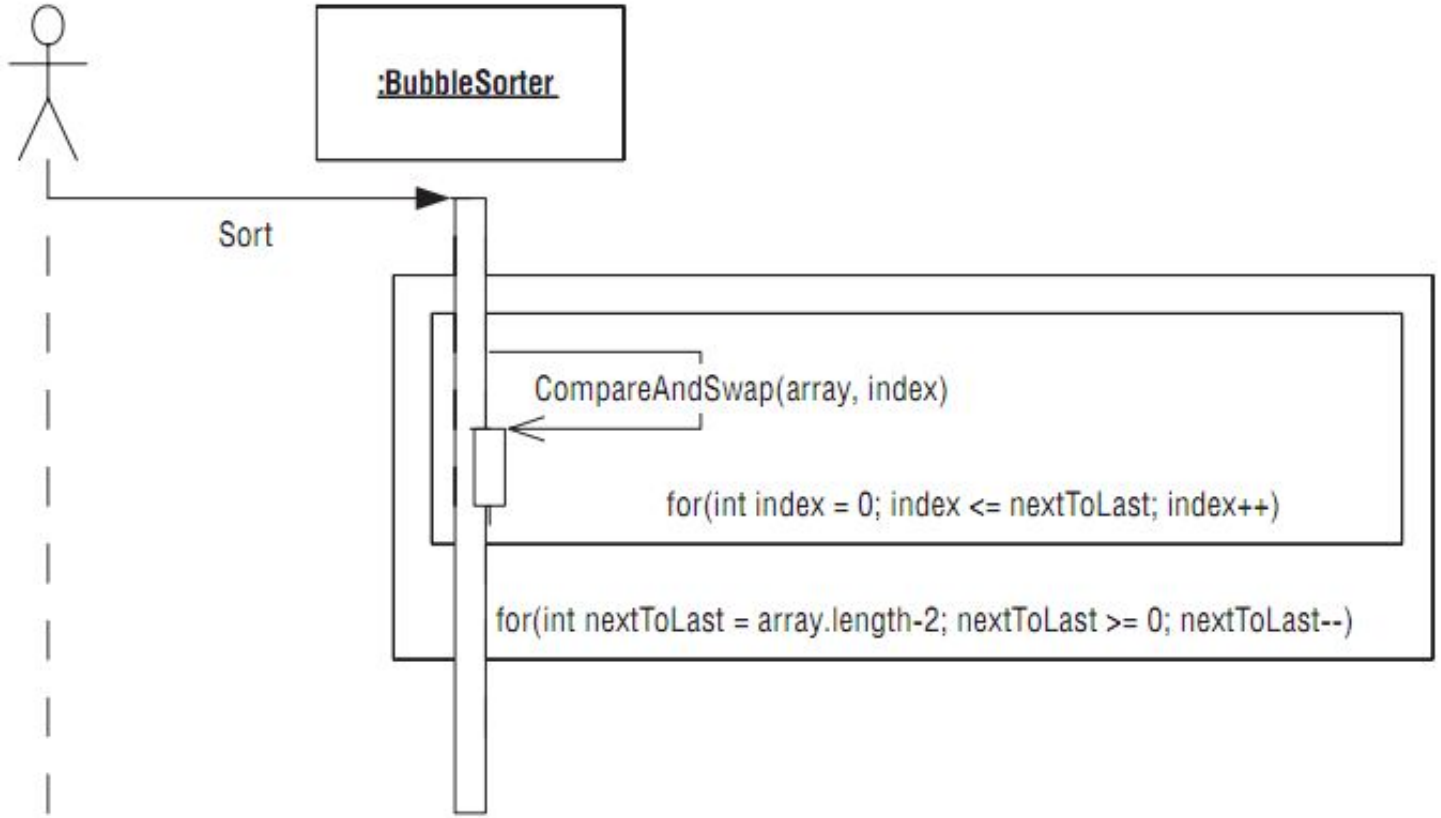


Диаграмма последовательности BubbleSorter

Карты

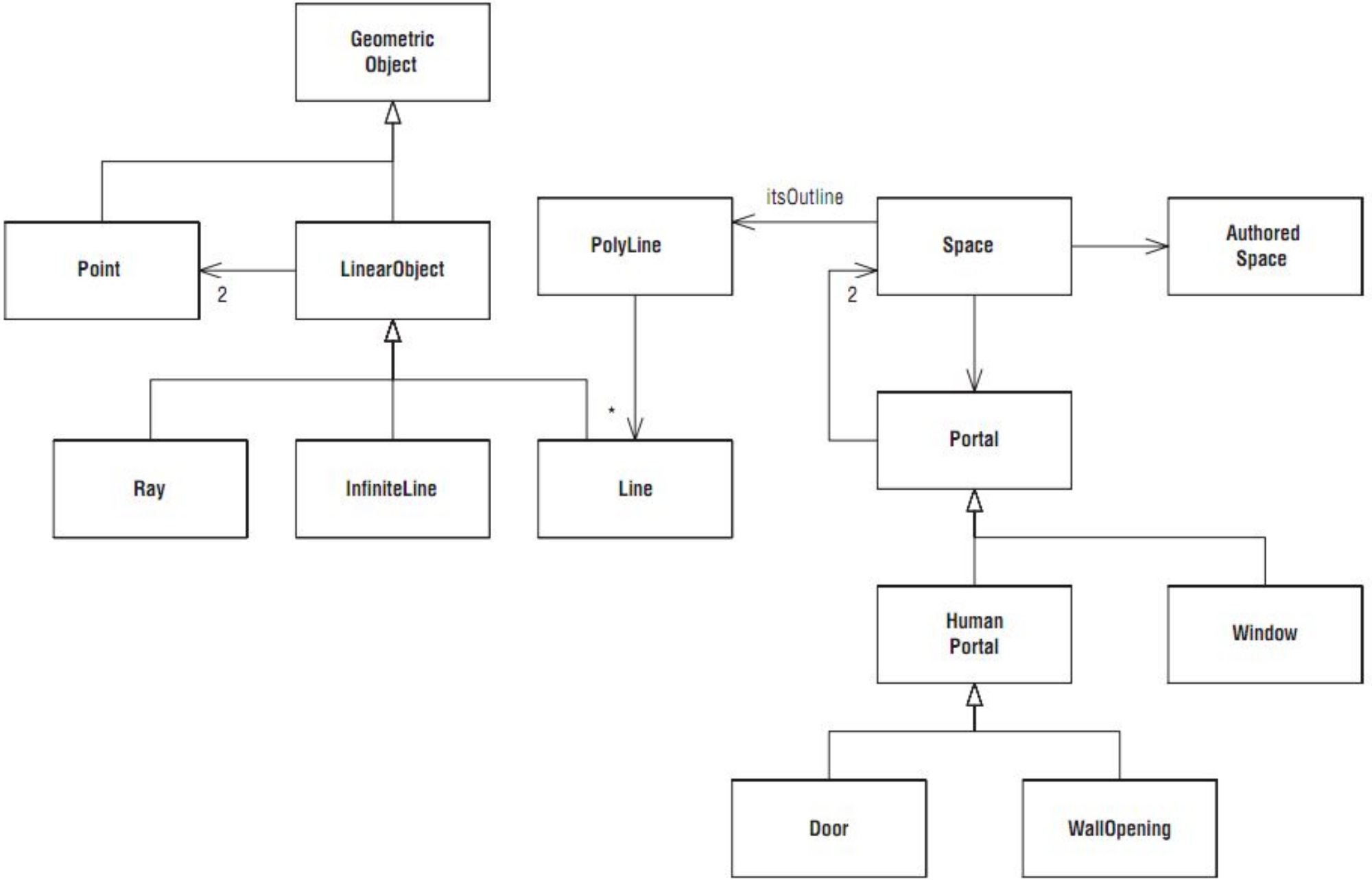
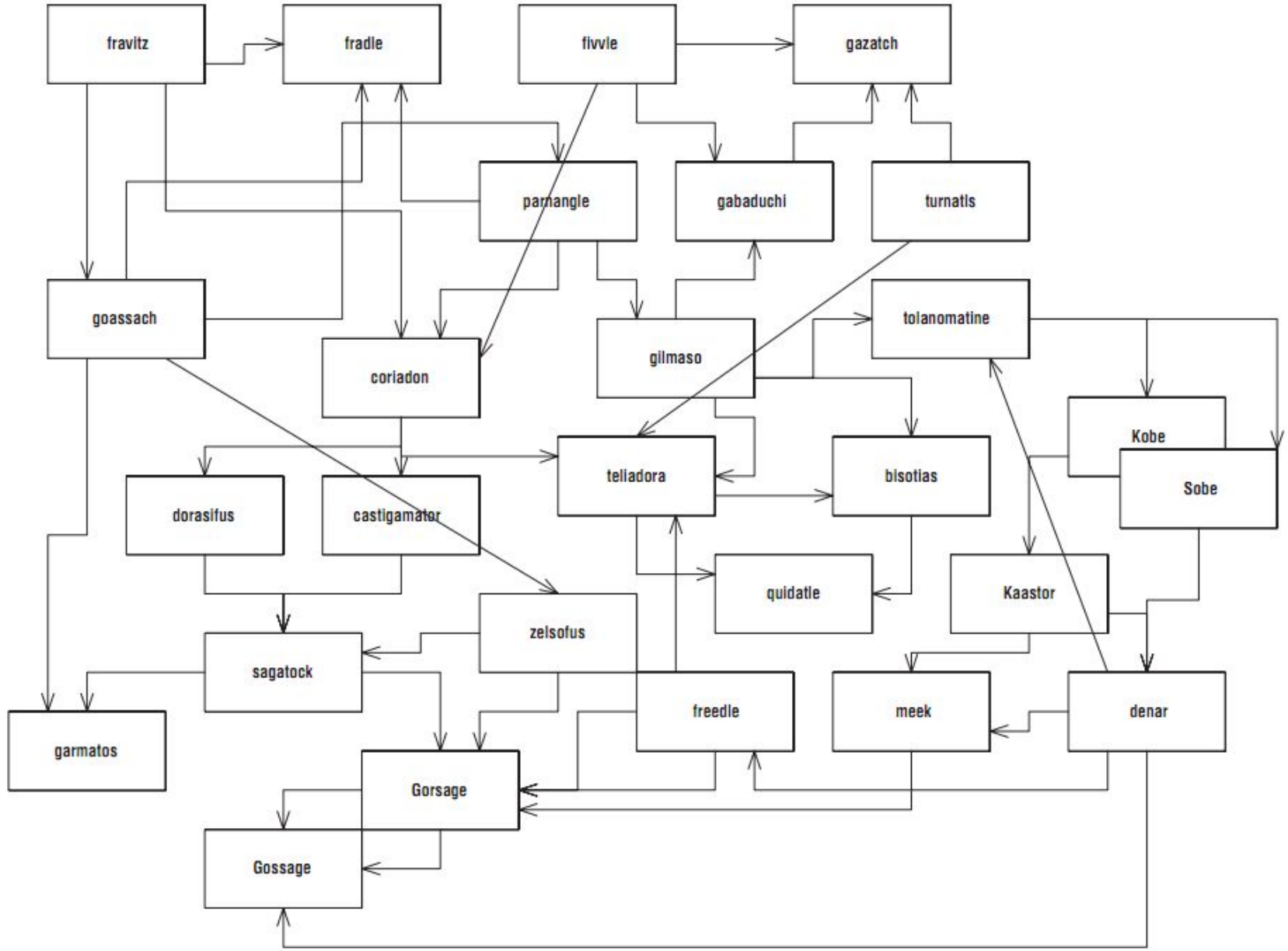
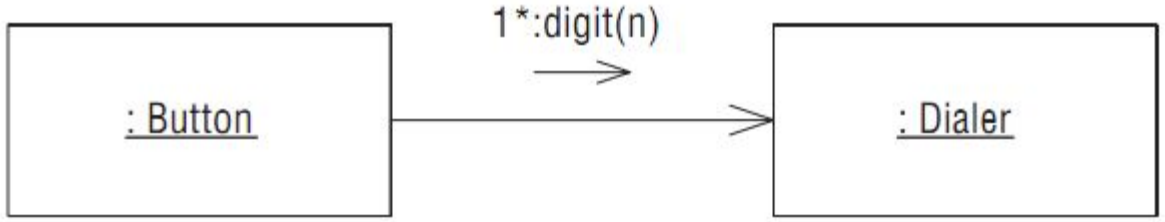


Диаграмма-карта

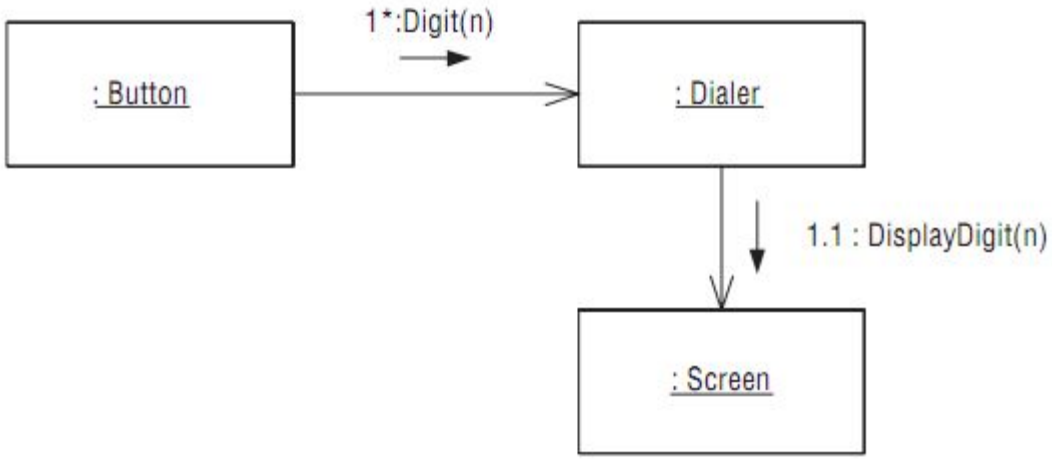
Финальная документация

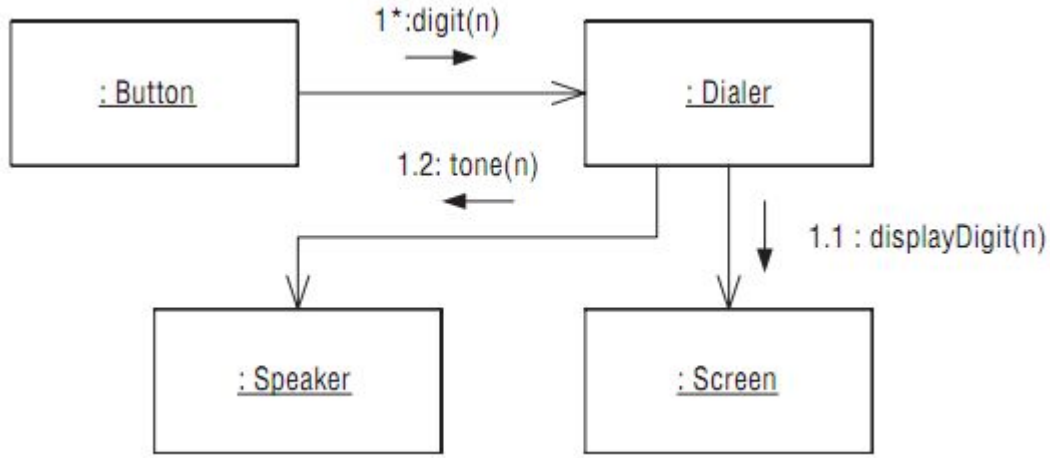


Плохой, но очень распространенный пример



Простая диаграмма последовательности





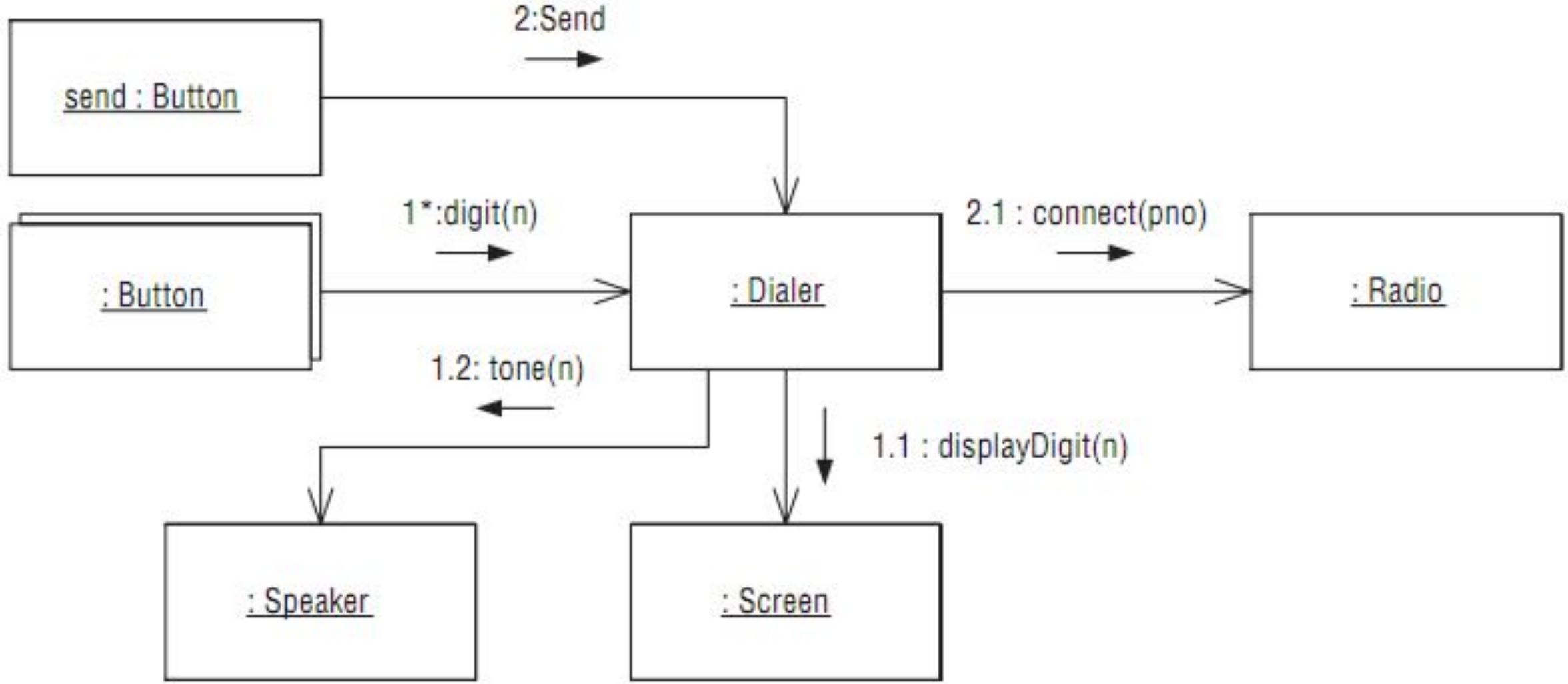


Диаграмма кооперации

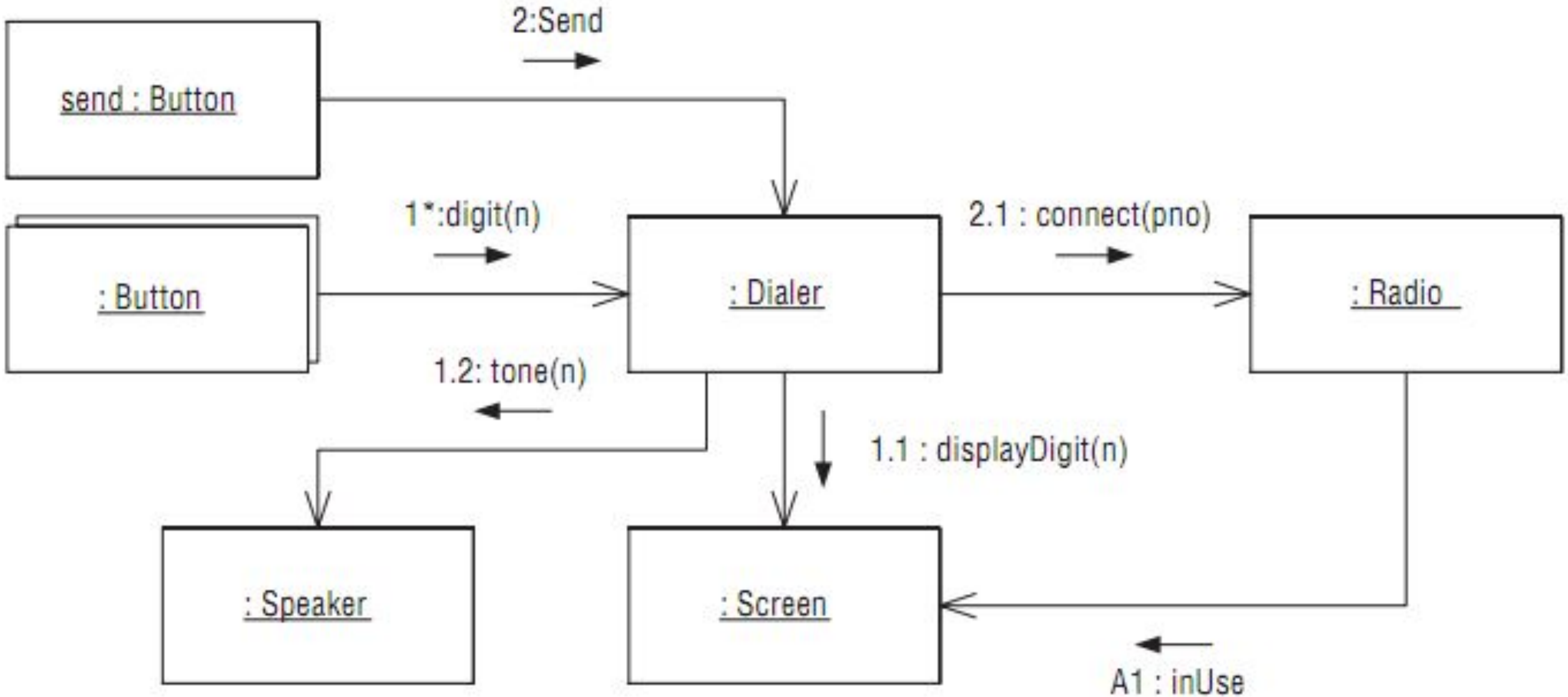


Диаграмма кооперации для мобильного телефона

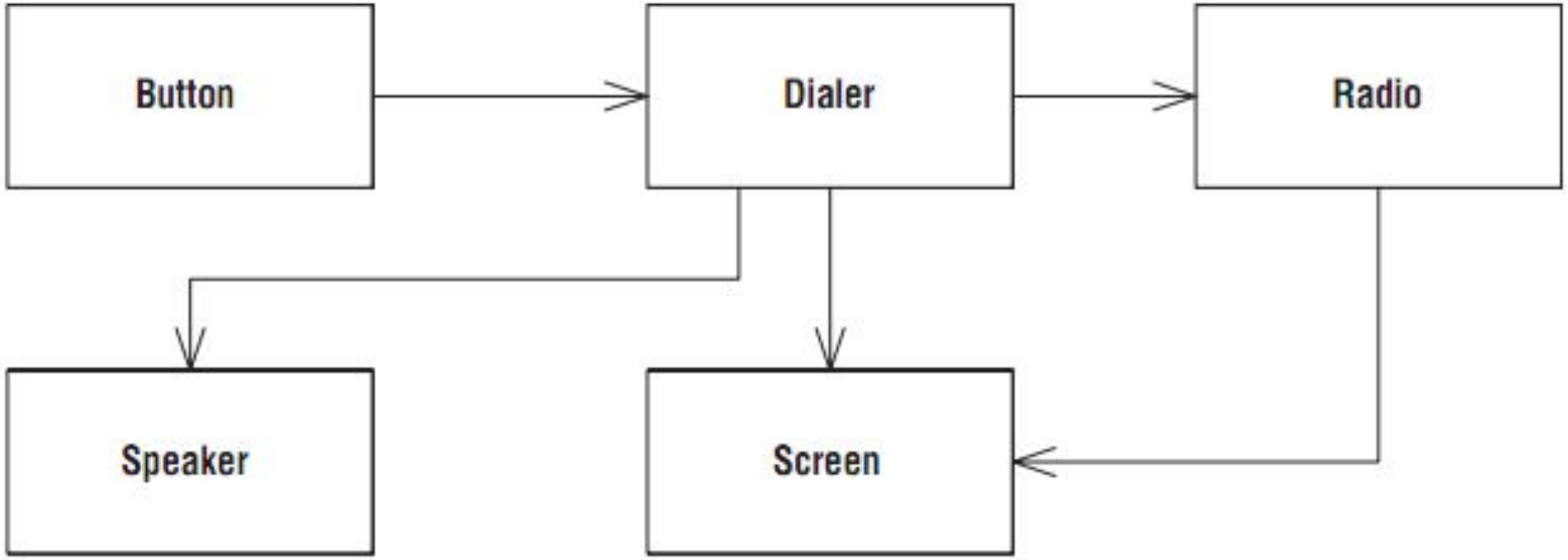
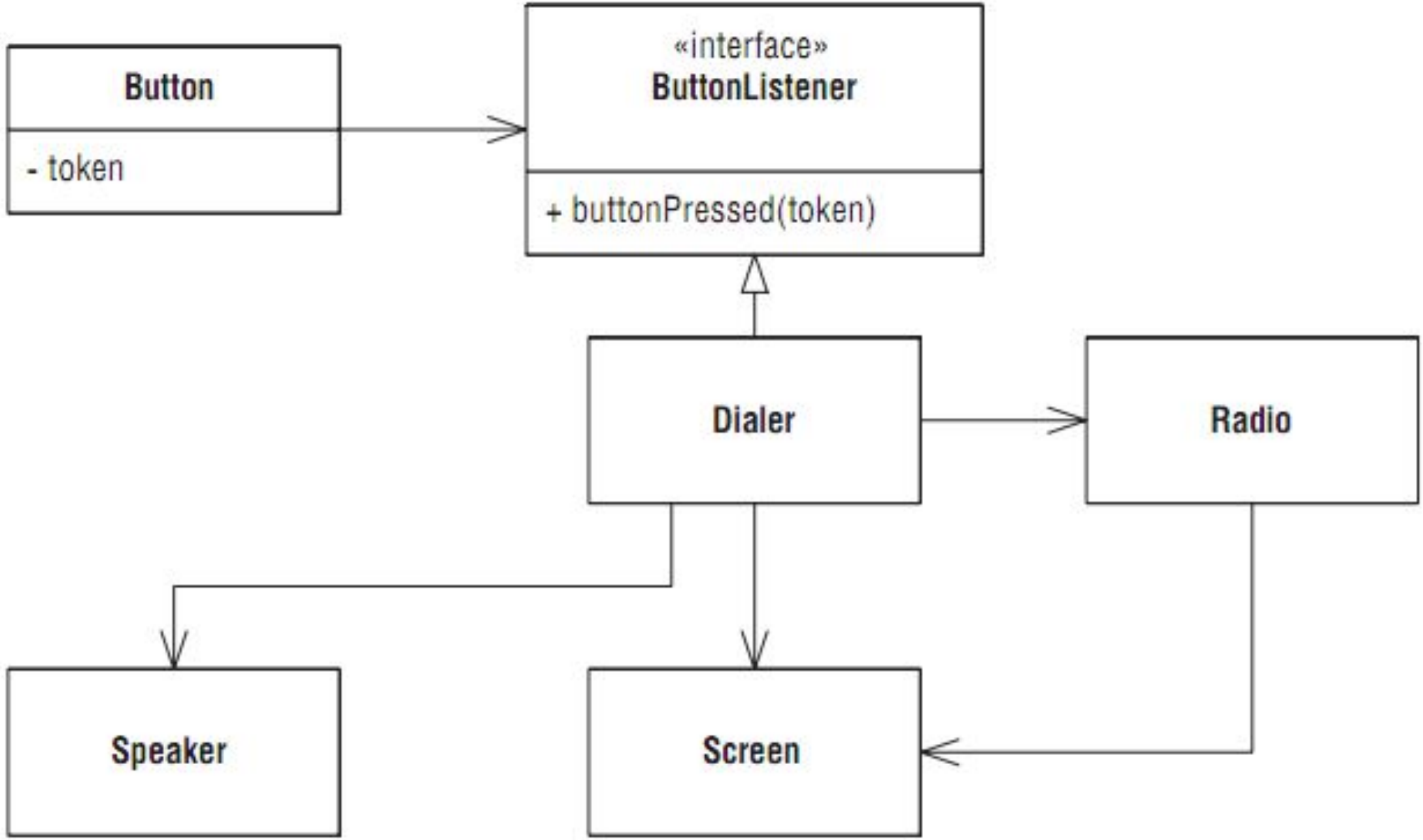
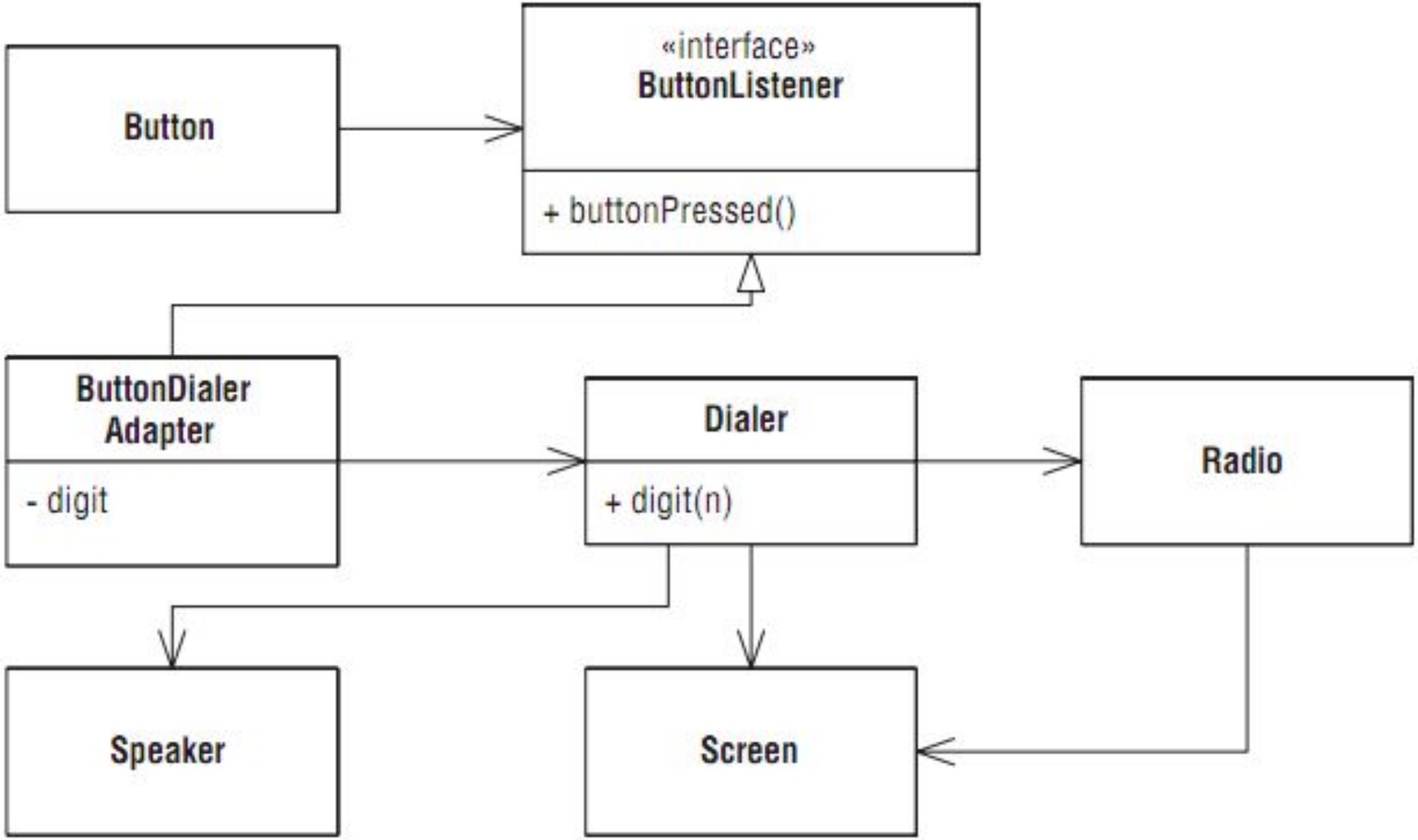


Диаграмма классов для мобильного телефона

```
public class Button
{
  private Dialer itsDialer;
  public Button(Dialer dialer)
  { itsDialer = dialer; }
}
```

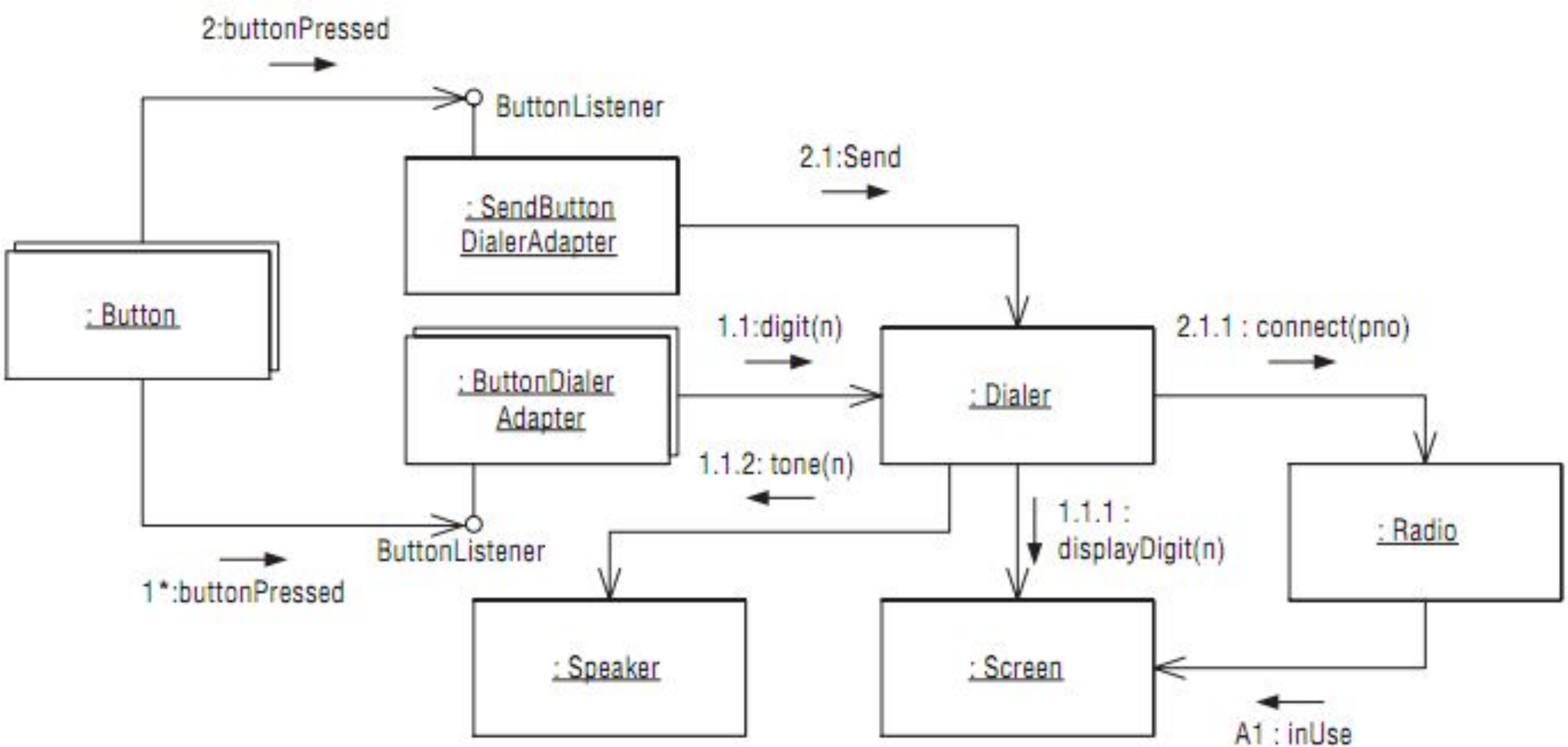


Изолирование Button от Dialer



Адаптация класса Button к классу Dialer

```
public class ButtonDialerAdapter : ButtonListener
{
    private int digit;
    private Dialer dialer;
    public ButtonDialerAdapter(int digit, Dialer dialer)
    {
        this.digit = digit;
        this.dialer = dialer;
    }
    public void ButtonPressed()
    {
        dialer.Digit(digit);
    }
}
```



Добавление адаптеров в динамическую модель