สายอักขระ

คือสายที่ไม่ได้เอาไปคำนวณ

จะเป็นตัวเลขหรือตัวอักษรก็ได้

หัวข้อ

- สายอักขระ
- คลาส String
- คลาส StringBuffer

สายอักขระ

สายฮักขระ

```
public class HelloWorld {

    public static void main(String[] args) {

        System.out.println("Hello, World! 2");

    }

}
```

ประโยชน์ของสายอักขระ

- บอกผลลัพธ์ของการคำนวณให้เราทราบ
- รายงานขั้นตอนการทำงานของโปรแกรม
- เก็บข้อมูลที่เป็นตัวอักษร ตัวเลข สัญลักษณ์

คลาส String

การใช้งานสตริง

- ประกาศตัวแปร

```
String name;
```

- กำหนดค่า

```
name = new String("Smith");
```

- แสดงผล

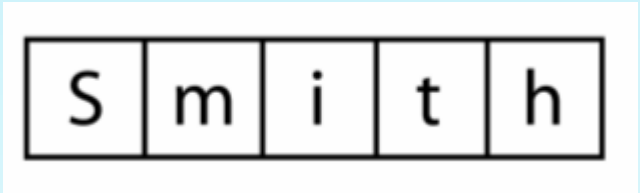
```
System.out.println(name);
```

เรฟเฟอร์เรนซ์ (Reference)

- int n
  - n เป็นข้อมูลชนิดจำนวนเต็ม
- String name
  - ไม่ได้หมายความว่า name เป็นวัตถุ String
  - แต่เป็นการบอกว่า name เป็นเรฟเฟอร์เรนซ์ (reference) หรือตัวที่ใช้อ้างอิงไปที่วัตถุ String

วัตถุและเรฟเฟอร์เรนซ์

- วัตถุ



- เรฟเฟอร์เรนซ์

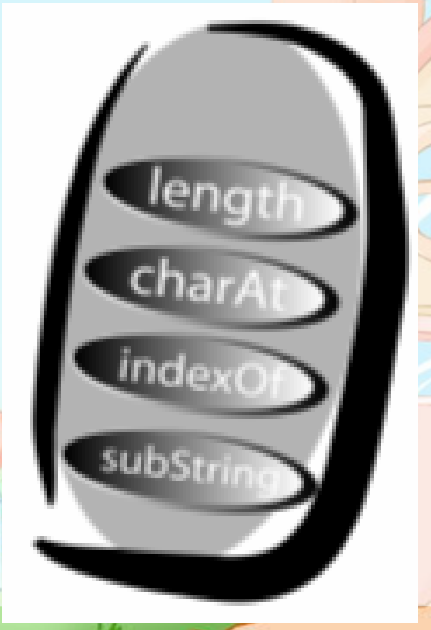
การสั่งงานสตริง

รูปแบบ

- เรฟเฟอร์เรนซ์.ข้อความ(สิ่งที่ส่งไปพร้อมกับข้อความ)

ตัวอย่าง

- name.length();
- name.charAt(1);



เมธอดที่น่าสนใจ

- length() ความยาวของสตริง
- charAt() ตัวอักษรในตำแหน่งที่กำหนด
  - ตัวอักษรตัวแรกคือตำแหน่งที่ 0
  - ตัวอักษรสุดท้ายคือ length() - 1
- indexOf() ตำแหน่งของสายอักขระในสตริง
- substring() สตริงที่อยู่ในช่วงที่กำหนด

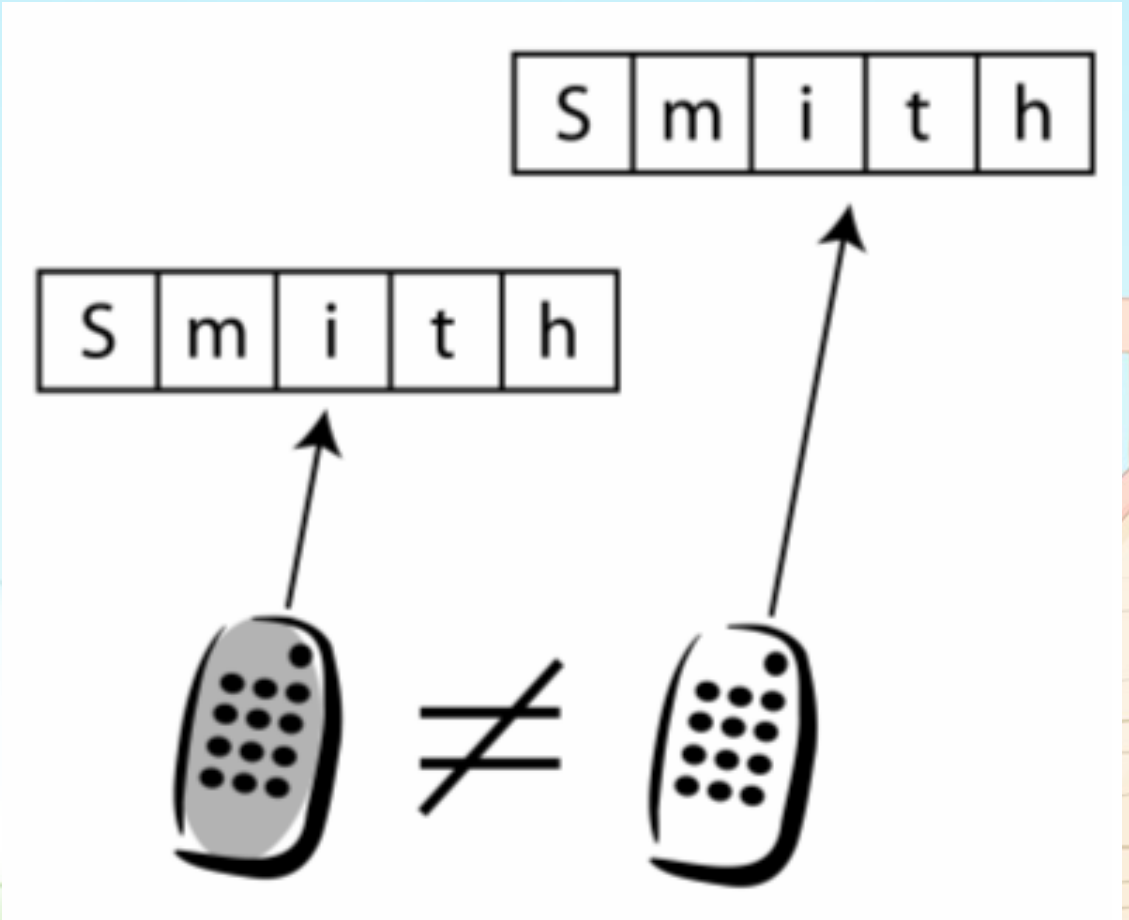
ตัวอย่างการใช้เมธอด

```
public class StringMessage {
    public static void main(String[] args) {
        String name = "Smith";
        System.out.println(name.length());
        System.out.println(name.charAt(2));
        System.out.println(name.indexOf("t"));
        System.out.println(name.substring(1, 3));
    }
}
```

การเปรียบเทียบสตริง (แบบผิดๆ)

```
public class StringCompareInCorrect {
    public static void main(String[]  args) {
        String name1 = new String(“Smith”);
        String name2 = new String(“Smith”);
        System.out.println(name1 == name2);
    }
}
```

การเปรียบเทียบสตริง (แบบผิดๆ)



การเปรียบเทียบสตริง

```
public class StringCompare {
    public static void main(String[]  args) {
        String name1 = new String(“Smith”);
        String name2 = new String(“Smith”);
        System.out.println(name1.equals(name2));
    }
}
```

การต่อสตริง

- เมธอด concat()
- เครื่องหมาย +
- เครื่องหมาย +=

เมธอด concat()

```
public class StringConcat1 {
    public static void main(String[] args) {
        String name = "Smith";
        String lastName = " Brown";
        String fullName = name.concat(lastname);
        System.out.println(fullName);
    }
}
```

เครื่องหมาย +

```
public class StringConcat2 {
    public static void main(String[] args) {
        String name = "Smith";
        String lastName = " Brown";
        String fullName = name + lastName;
        System.out.println(fullName);
    }
}
```

เครื่องหมาย +=

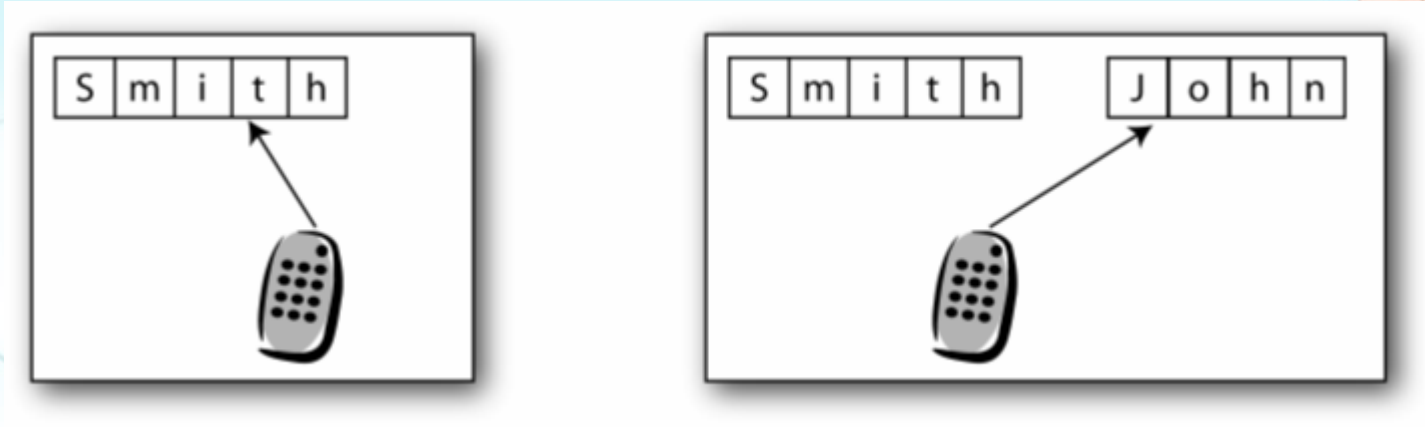
```
public class StringAppend {
    public static void main(String[]  args) {
        String name = "Smith";
        String lastName = " Brown";
        name += lastName;
        System.out.println(name);
   }
}
```

ความมั่นคงของสตริง (Immutability)

- ไม่สามารถเปลี่ยนแปลงค่าของสายอักขระได้เลยหลังจากที่เราได้สร้างมันขึ้นมา
- ในบรรดาข้อความ (message) ที่วัตถุ String รู้จักนั้น ไม่มีข้อความใดเลยที่ทำให้สายอักขระเปลี่ยนแปลงได้
  - length()
  - substring()

สตริงไม่เปลี่ยนแปลง

```
String str;
str = new String("Smith");
str = new String("John");
```



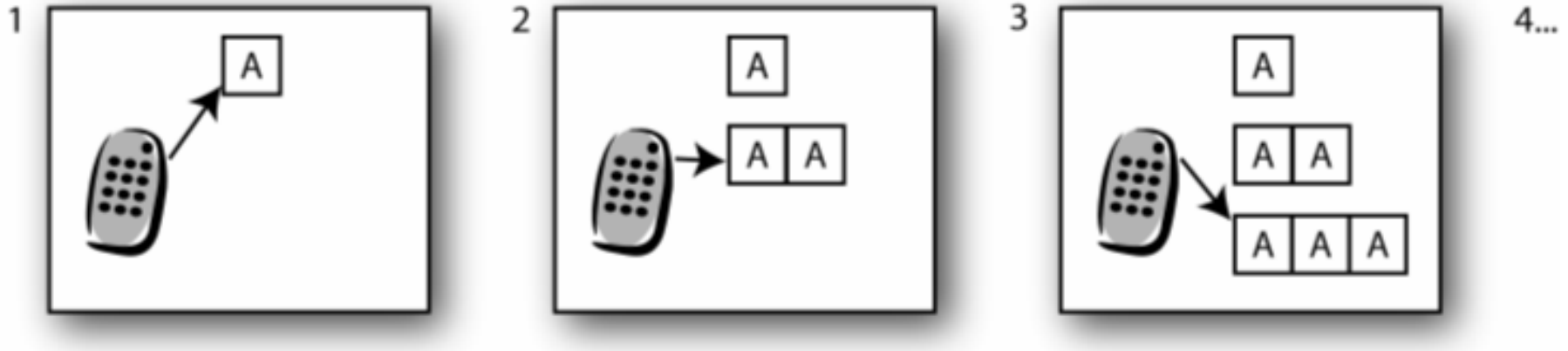
สตริงไม่เปลี่ยนแปลง

```
String str = "Smith";
System.out.println(str.toUpperCase());
System.out.println(str);
```

สตริงไม่เปลี่ยนแปลง

```
public class StringDeficiency {
    public static void main(String[] args)
    {
        String str = "A";
        for (int i = 0; i < 100; i++)
            str += "A";
        System.out.println(str);
    }
}
```

สตริงไม่เปลี่ยนแปลง



คลาส StringBuffer

ข้อดีของ StringBuffer

- ประหยัดทั้งหน่วยความจำและเวลาในการประมวลผล
  - วัตถุ StringBuffer สามารถเปลี่ยนแปลงค่าในสายอักขระที่มันนำเสนอได้เอง โดยไม่ต้องสร้างวัตถุขึ้นมาใหม่

เมธอดที่น่าสนใจ

- append() การต่อสายอักขระ
- insert() การแทรกสายอักขระ
- delete() การลบสายอักขระย่อย

ตัวอย่างการใช้เมธอด

```
StringBuffer sb = new StringBuffer("John");
System.out.println(sb);

sb.append(" Hunter");
System.out.println(sb);

sb.insert(4, "y");
System.out.println(sb);

sb.delete(2, 4);
System.out.println(sb);
```

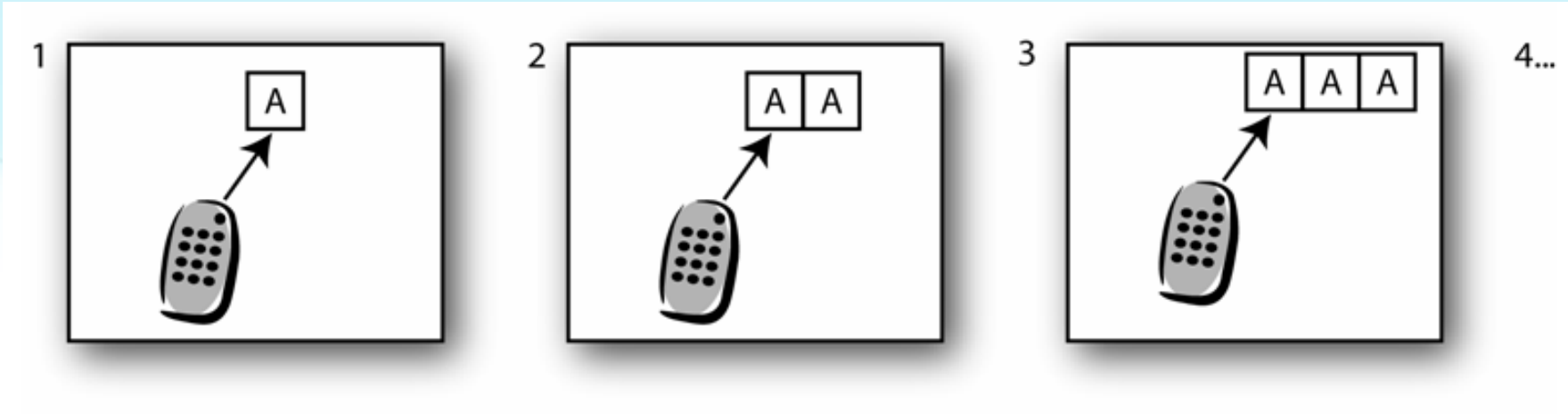
ทดลองเปลี่ยนแปลง StringBuffer

```
public class StringBufferAppend {
    public static void main(String[] args) {
        StringBuffer sb;
        sb = new StringBuffer("A");

        for (int i = 0; i < 100; i++)
            sb.append("A");

        System.out.println(sb);
    }
}
```

ทดลองเปลี่ยนแปลง StringBuffer



StringBuilder

- มีเมธอดที่เหมือนกับ StringBuffer
- ทำงานได้เร็วกว่า StringBuffer
- StringBuffer ปลอดภัยสำหรับการทำงานในแบบ multiple threads

Java Doc

เอกสารอ้างอิง

แหล่งเอกสาร

- http://java.sun.com/j2se/1.5.0/docs/api
- http://java.sun.com/j2se/1.5.0/docs/api/java/lang/String.html
- http://java.sun.com/j2se/1.5.0/docs/api/java/lang/StringBuffer.html

สรุป

สรุป

- สายอักขระมีไว้เพื่อการแสดงผลข้อความ
- สายอักขระได้ถูกนำเสนอด้วยวัตถุ String และ StringBuffer
- เราสามารถใช้งานวัตถุได้โดยผ่านทางเรฟเฟอร์เรนซ์
- เรฟเฟอร์เรนซ์เปรียบเสมือนรีโมทคอนโทรล
- วัตถุเปรียบได้กับสิ่งที่รีโมทนั้นควบคุมอยู่
- การส่งข้อความหาวัตถุเปรียบได้กับการกดปุ่มรีโมทคอนโทรล

สรุป

- การเปรียบเทียบสตริงทำได้โดยการส่งข้อความ equals( )
- การต่อสตริงนอกจากจะใช้การส่งข้อความ concat() แล้ว ยังสามารถใช้เครื่องหมาย + ได้อีกด้วย
- สายอักขระของวัตถุ String ไม่สามารถถูกเปลี่ยนแปลงได้
- ถ้าต้องการเปลี่ยนแปลงค่าในสายอักขระบ่อยๆ ควรใช้วัตถุ StringBuffer