Strings

StringBuilder

Creating a StringBuilder

- StringBuilder is a **mutable** class which contains a String
  - it has many useful methods for manipulating the strings

```
StringBuilder name = new StringBuilder("John Wayne");
```

- some methods work in the identical way as with a normal String
  - `substring()`, `indexOf()`, `length()` and `charAt()`

```
// append()
StringBuilder name = new StringBuilder("John");
name.append("Wayne");
System.out.println(name);
  => JohnWayne          // StringBuilder is mutable!

// chaining with append()
name.append(1).append(true);
System.out.println(name);
  => JohnWayne1true     // all arguments are converted to String
```

```
// insert()
StringBuilder name = new StringBuilder("John Wayne");
                                       5
name.insert(5, "D. ");
System.out.println(name);
  => John D. Wayne


// chaining with insert
StringBuilder name = new StringBuilder("John Wayne");
name.insert(5, "D. ").insert(6, "A");
                      6
  // John D. Wayne -> John DA. Wayne
System.out.println(name);
  => John DA. Wayne
```

```
// delete()
                                                  1   4
StringBuilder sb1 = new StringBuilder("abcdef");
System.out.println(sb1.delete(1, 4));
  => aef    // deletes from 1 to 4 (excluded)


// deleteCharAt()
StringBuilder sb2 = new StringBuilder("abcdef");
System.out.println(sb2.deleteCharAt(2));
  => abdef   // c is deleted
System.out.println(sb2.deleteCharAt(6));
  => StringIndexOutOfBoundsException
```

```
// replace()
                                                  1  3
StringBuilder sb = new StringBuilder("abcdef");
sb.replace(1, 3, "JOHN");
  // removes characters from index 1 to 3 (excluded) and inserts new string
  // in this case, 'b' and 'c' will be removed and "JOHN" will be inserted
  // notice the different syntax that in String method replace() !!
System.out.println(sb);
  => aJOHNdef


// if final index is too large, replace goes through the end (no exception!)
StringBuilder name = new StringBuilder("John Wayne");
name.replace(5, 100, "Doe");
System.out.println(name);
  => John Doe
```

```
// reverse()

StringBuilder sb = new StringBuilder("LUKA");

sb.reverse();

System.out.println(sb);

  => AKUL



// toString()

StringBuilder sb = new StringBuilder("John Wayne");

String str = sb.toString();
```

```
// StringBuilder doesn't implement equals() method!!
// i.e. equals() is same as ==

StringBuilder sbName1 = new StringBuilder("John Wayne");
StringBuilder sbName2 = new StringBuilder("John Wayne");
System.out.println(sbName1 == sbName2);
  => false
System.out.println(sbName1.equals(sbName2));
  => false
// if we want to compare content we have to convert it back to String:
System.out.println(sbName1.toString().equals(sbName2.toString()));
  => true
```

```
// substring() returns a String and doesn't change the StringBuilder

StringBuilder name = new StringBuilder("John Wayne");

name.substring(2, 6);

System.out.println(name);

  => John Wayne


// fix

String subName = name.substring(2, 6);

System.out.println(subName);

  => hn W
```