Static Members
What's This?
Documentation
Javadoc
Encapsulation

TA session 3

Static members

Class memory

Object memory

class SecretSpy

```
private static spyList

public
static broadcastMsgToAllSpies
```

SecretSpy hershale

```
private int serialNumber

private receiveMsg(msg)
public sendMsgToSpy(spy)
public sendMsgToSpy(serial)
```

Static members

```
public void sendMsgToSpy(SecretSpy spy, String msg) {
    spy.receiveMsg(msg);
}
```

class SecretSpy

```
private static spyList

public
static broadcastMsgToAllSpies
```

SecretSpy hershale

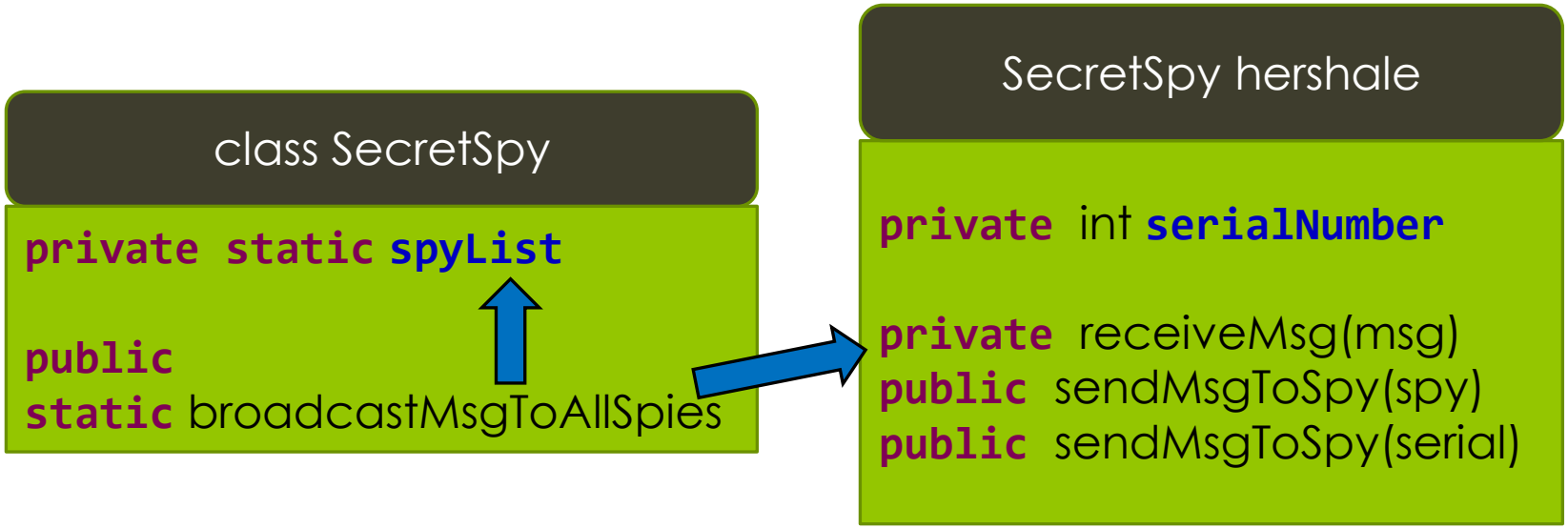
```
private int serialNumber

private receiveMsg(msg)
public sendMsgToSpy(spy)
public sendMsgToSpy(serial)
```

Static members

```
public static void broadcastMsgToAllSpies (String msg) {
    for(SecretSpy spy : spyList)
        spy.receiveMsg(msg);
}
```

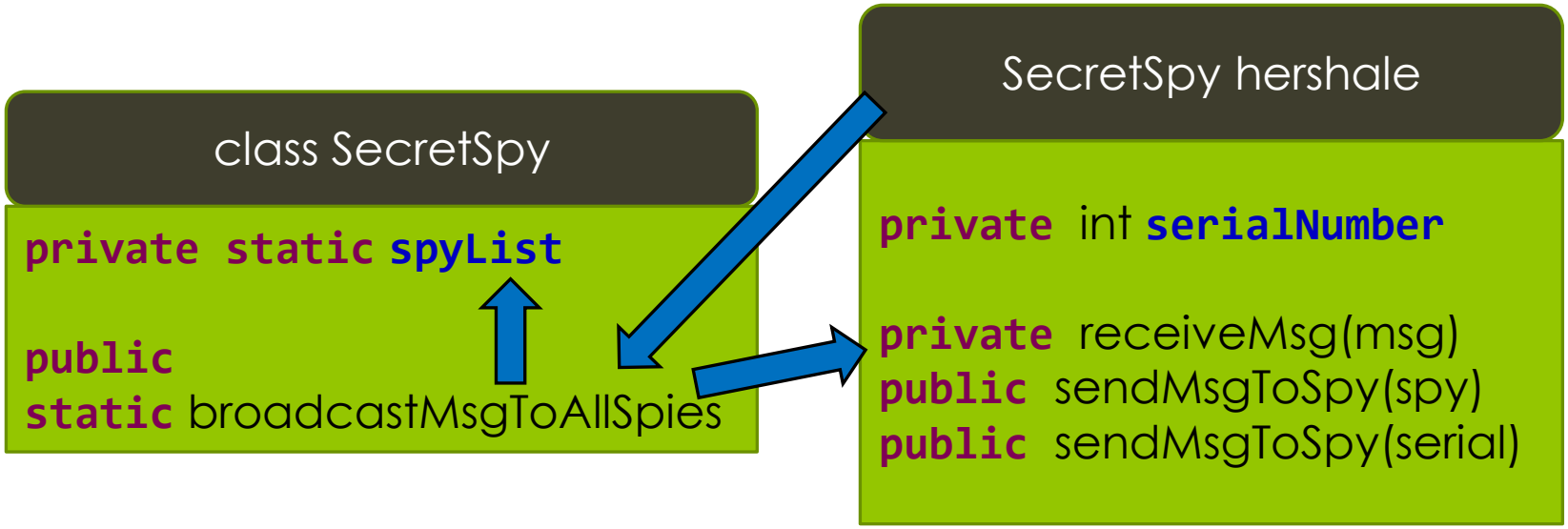
```
SecretSpy.broadcastMsgToAllSpies(msg)
```



Static members

```
public static void broadcastMsgToAllSpies (String msg) {
    for(SecretSpy spy : spyList)
        spy.receiveMsg(msg);
}
```

```
hershale.broadcastMsgToAllSpies(msg)
```



Static members

```
public void sendMsgToSpy (int serial, String msg) {
    for(SecretSpy otherSpy : spyList) {
        if(otherSpy.serialNumber == serial) {
            sendMsgToSpy(otherSpy,msg);
            break;
        }
    }
}
```

Why not receiveMsg?

class SecretSpy

```
private static spyList

public
static broadcastMsgToAllSpies
```

SecretSpy hershale

```
private int serialNumber

private receiveMsg(msg)
public sendMsgToSpy(spy)
public sendMsgToSpy(serial)
```

This: Java's "self"

- An object's reference to itself.
- Main uses:
  - Reference to shadowed fields
  - Explicit constructor invocation
  - As a method parameter

This: Java's "self"

- Reference to shadowed fields

```
public class ComplexNumber {
    private double real, img;

    public ComplexNumber(final double real, final double img){
        this.real = real;
        this.img  = img;
    }
}
```

This: Java's "self"

- Explicit constructor invocation

```
public ComplexNumber(final double real, final double img){
    this.real = real;
    this.img  = img;
}

public ComplexNumber(final ComplexNumber other){
    this(other.real, other.img);
}
```

This: Java's "self"

- As a method parameter

```
public class ComplexNumber {
    private static ComplexNumber sumOfAllComplexNumbers =
                                        new ComplexNumber();
    private double real, img;

    public ComplexNumber(final double real, final double img){
        this.real = real;
        this.img  = img;
        sumOfAllComplexNumbers.add(this);
    }
    private ComplexNumber(){
        real = 0;
        img = 0;
    }
}
```

What if we used
new ComplexNumber(0,0)?

Code readability

- The user doesn't see your code. Why does it have to be readable?.
  - It doesn't.
  - Debugging is rarely needed.
  - Code is never read by colleagues or future developers.
  - Why would you want to read your own code again?

Code readability

- What makes code readable?
  - The code!
    - Intuitiveness of the flow
    - Names
    - Organization
    - Conventions

Code readability

```
private static final int MIN_ALIGNED_DIGITS = 4;
private static boolean isContainMinAlignedDigits(String input) {
    int count = 0;
    for(int i = 0 ; i < input.length() &&
                    i < 10 &&
                    count < MIN_ALIGNED_DIGITS ; i++) {
        if(input.charAt(i) == Character.forDigit(i, 10))
            count++;
    }
    return count == MIN_ALIGNED_DIGITS;
}
```

3 -> '3'

Code readability

- Sometimes it's not enough
  - We need natural language

Now that's better!

```
//minimum aligned digits in a string
private static final int MIN_ALIGNED_DIGITS = 4;
//checks if input contains enough aligned digits
private static boolean isContainMinAlignedDigits(String input) {
    int count = 0; //counter for aligned digits
    for(int i = 0 ; i < input.length() &&
                    i < 10 && // while i is still a digit
                    count < MIN_ALIGNED_DIG
        if(input.charAt(i) == Character.fo
            count++; // found an aligned d
    }
    //return if found enough aligned digi
    return count == MIN_ALIGNED_DIGITS;
}
```

Assume the reader knows how to read

```
//minimum aligned digits in a string
private static final int MIN_ALIGNED_DIGITS = 4;
//checks if input contains enough aligned digits
private static boolean isContainMinAlignedDigits(String input) {
    int count = 0; //counter for aligned digits
    for(int i = 0 ; i < input.length() &&
                    i < 10 &
                    count <
        if(input.charAt(i) == Character.forDigit(i, 10))
            count++; // found an aligned digit
    }
    //return if found enough aligned digits
    return count == MIN_ALIGNED_DIGITS;
}
```

Assume the reader isn't retarded

Assume the reader knows Java

Create a jargon and explain it

```
//minimum aligned digits in a string
private static final int MIN_ALIGNED_DIGITS = 4;
//checks if input contains enough aligned digits
private static boolean isContainMinAlignedDigits(String input) {
    int count = 0; //counter for aligned digits
    for(int i = 0 ; i < input.length() &&
                    i < 10 && // while i is still a digit
                    count < MIN_ALIGNED_DIGITS ; i++) {
        if(input.charAt(i) == Character.forDigit(i, 10))
            count++; // found an aligned digit
    }
    //return if found enough aligned digits
    return count == MIN_ALIGNED_DIGITS;
}
```

Do NOT assume he's not cursing

```java
// See isContainMinAlignedDigits
private static final int MIN_ALIGNED_DIGITS = 4;
/*
 * An "aligned digit" is a digit that appears in an index equal
 * to the digit. For Example:
 * a1aaa: 1 is an aligned digit
 * 01aa4b: contains 3 aligned digits
 */
private static boolean isContainMinAlignedDigits(String input) {
    int count = 0;
    for(int i = 0 ; i < input.length() &&
                    i < 10 &&
                    count < MIN_ALIGNED_DIGITS ; i++) {
        if(input.charAt(i) == Character.forDigit(i, 10))
            count++;
    }
    return count == MIN_ALIGNED_DIGITS;
}
```

Better.

Documentation: even better

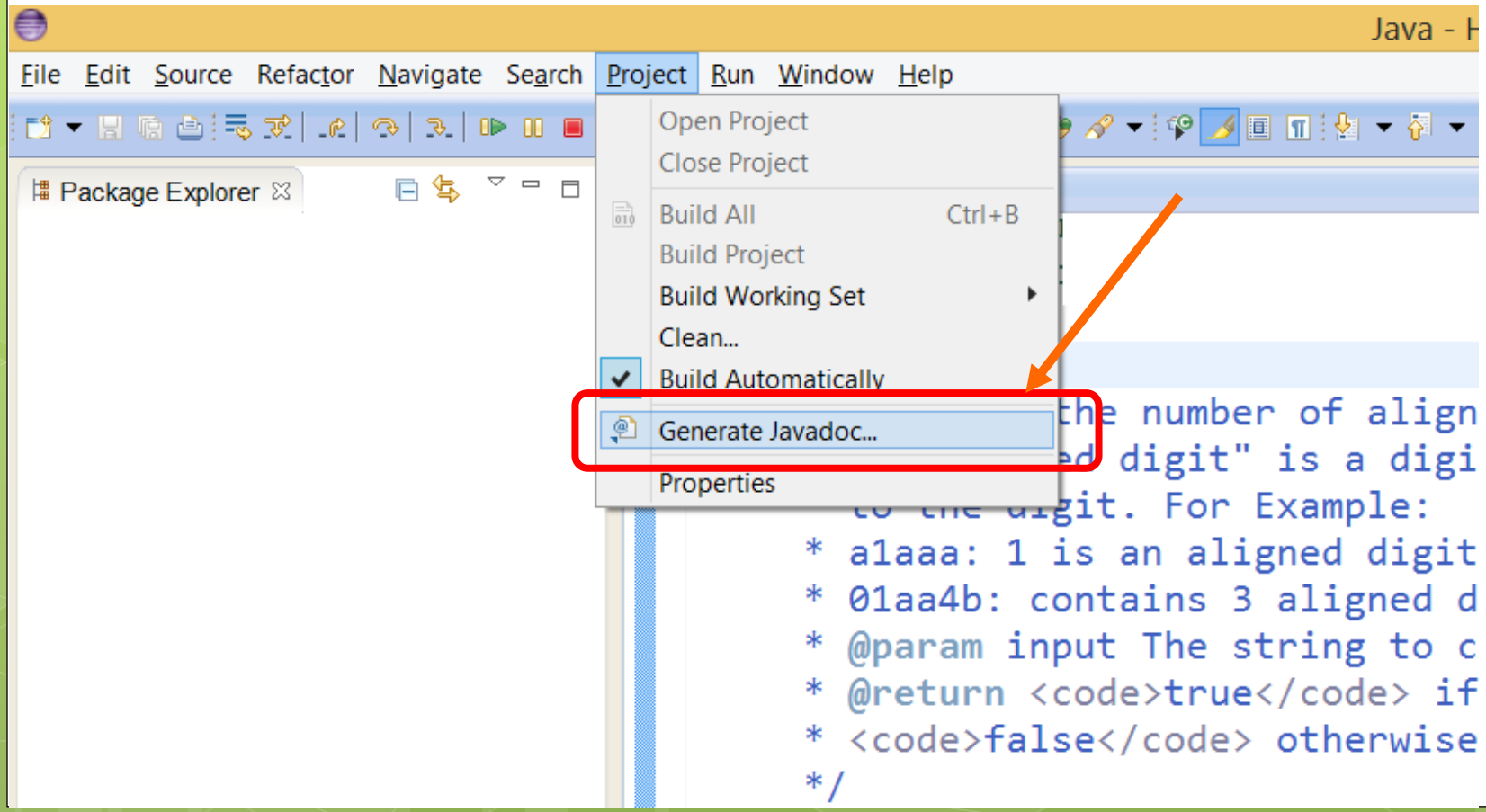
```
/**
 * An "aligned digit" is a digit that appears in an index equal
 * to the digit. For Example:
 * a1aaa: 1 is an aligned digit
 * 01aa4b: contains 3 aligned digits
 *
 * @param input: The string to check for aligned digits.
 * @return True if input contains at least MIN_ALIGNED_DIGITS,
 * false otherwise.
 */
```

Do users of the class have to read code?

Javadoc.

Javadoc

- Generates HTML documentation from standardized documentation syntax.



Javadoc

```
/**
 * An "aligned digit" is a digit that appears in an index equal
 * to the digit. For Example:
 * a1aaa: 1 is an aligned digit
 * 01aa4b: contains 3 aligned digits
 *
 * @param input: The string to check for aligned digits.
 * @return True if input contains at least MIN_ALIGNED_DIGITS,
 * false otherwise.
 */
```

Javadoc syntax

Javadoc

Class HelloClass

java.lang.Object
    HelloClass

```
public class HelloClass
extends java.lang.Object
```

Field Summary

Fields

Modifier and Type	Field and Description
`private static int`	`MIN_ALIGNED_DIGITS` See isContainMinAlignedDigits

Constructor Summary

Constructors

Constructor and Description
`HelloClass()`

Method Summary

Methods

Modifier and Type	Method and Description
`private static boolean`	`isContainMinAlignedDigits(java.lang.String input)` Compares the number of aligned digits in a string to MIN_ALIGNED_DIGITS.
`private static boolean`	`isStringPalindromeOrHasMirrorDigits(java.lang.String str)`
`static void`	`main(java.lang.String[] args)`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Javadoc

isContainMinAlignedDigits

```
private static boolean isContainMinAlignedDigits(java.lang.String input)
```

Compares the number of aligned digits in a string to MIN_ALIGNED_DIGITS.
An "aligned digit" is a digit that appears in an index equal to the digit. For Example:
a1aaa: 1 is an aligned digit
01aa4b: contains 3 aligned digits

Parameters:

`input` - The string to check for aligned digits.

Returns:

`true` if input contains at least MIN_ALIGNED_DIGITS, `false` otherwise.

Encapsulation

- A capsule is
  - A black box.
  - Self contained.
- Modifications are transparent.

- Make your code a capsule.

Encapsulation: via API

- Carefully define what you need the class for.
- Your API should:
  - Meet the needs.
  - Nothing more.
- Hide everything else.

Encapsulation

```
public double real, img;
```

```
private double real, img;
public double getRealPart(){...
public double getImgPart(){...
public void setRealPart(double real){...
public void setImgPart(double img){...
```

Wow. That's silly.

What is this hiding?

In a Free World

1

```
public class ComplexNumber {
    public double real, img;
    // add and multiply methods
}
```

In a Free World

- Wait, I need polar coordinates as well.
- Free access: changing “real” doesn’t change “angle”.
- Let’s add a class.

```
public class PolarComplexNumber {
    private double real, img;
    private double norm, angle;

    public void setReal(double real){
        this.real = real;
        updatePolarCoords();
    }

    private void updatePolarCoords() { ... }
}
```

2

non-trivial setter

Users update code.

In a Free World

- Wait, Cartesian coordinates slow me down.
- But they already use "`setReal()`".
- Supply a faster class.

3

```
public class FastPolarComplexNumber {
    public double norm, angle;
    // add and multiply methods
}
```

Users update code.

In a Free World

- 3 similar classes.
- users update their a** off.
- Users have incompatible code.
- New user: Holy moly. Which class do I need?
- Complex-numbers-related code is all over the place.

Encapsulation

```
private double real, img;
public double getRealPart(){ ... }
public double getImgPart() { ... }
public void setRealPart(double real){...
public void setImgPart(double img){...
```

- No, wait. Why would a user need to set real/img?
- Allowing it:
  - Is lengthening the API
  - Limits our further development.

Encapsulation

- Wait, I need polar coordinates.

```
private double real, img, norm, angle;

public void setAngle(double angle){
    this.angle = angle;
    updateCartezCoords();
}

private void updateCartezCoords() { ... }
```

extending the API is acceptable

Users bake in the sun.

Encapsulation

- Wait. I don't need Cartesian coordinates at all.

```
private double norm, angle;
public double getRealPart(){ return norm*Math.cos(angle); }
public double getImgPart() { return norm*Math.sin(angle); }
```

Users have beauty naps.

Encapsulation

- Even lazy users experience increased performance.
- User code compatibility.
- Friendly API.
- Easy maintenance.