Chapter 6
A closer look at methods and classes

Based on the course literature:

Java: A beginner’s guide

Sixth Edition

Herbert Schildt

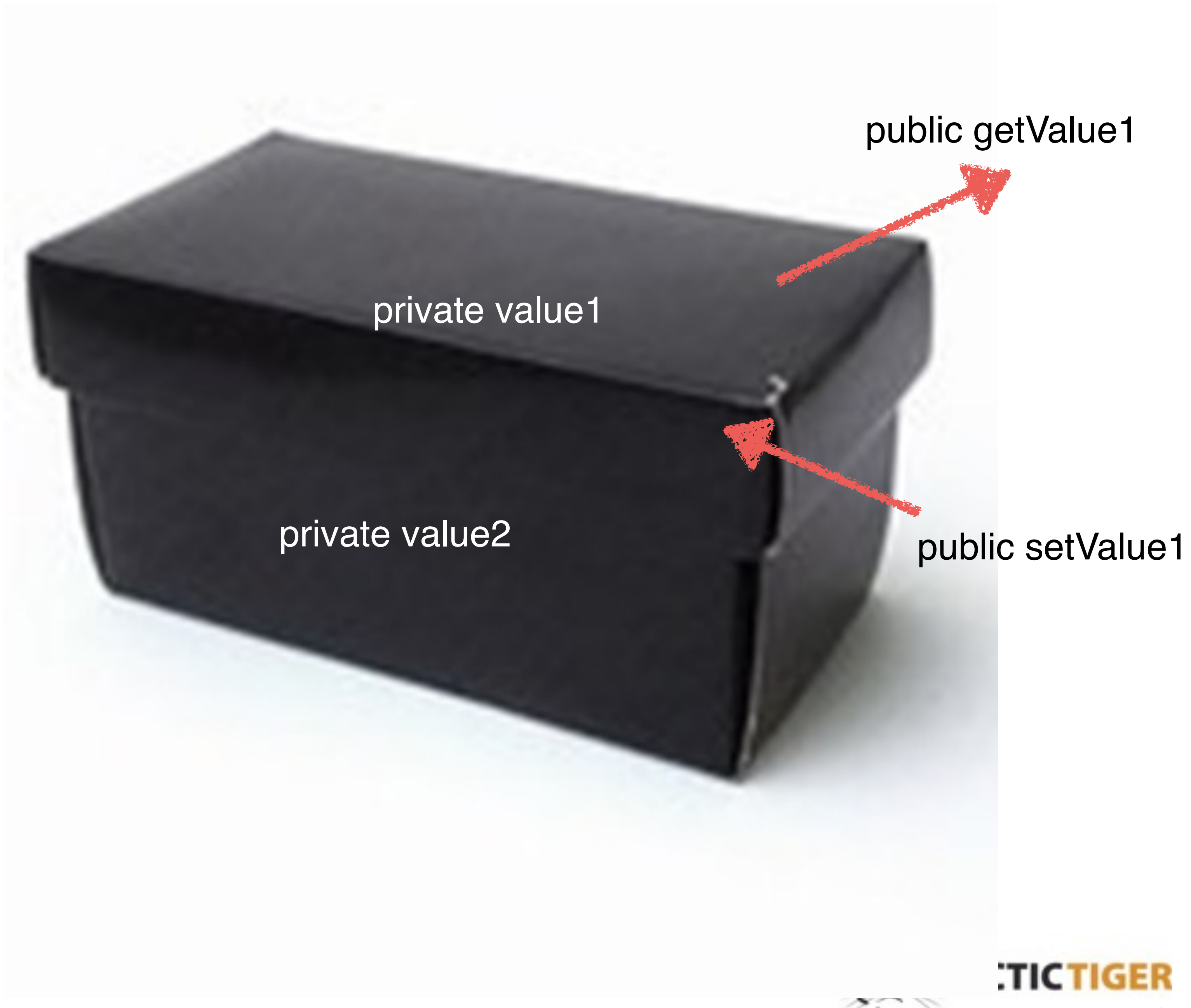
What we'll cover

- Control access to members
- Pass and return objects to and from methods
- Overload methods and constructors
- Use recursion
- Apply static
- Use inner classes
- Use varargs

Controlling access to members

- public members:
  can be accessed outside of the class the member is defined within.

- private members:
  can only be accessed inside of the class the member is defined within.


public getValue1
private value1
private value2
public setValue1

default

```
// A public class with a public method
class User {
	int getNumber(){
		return 3;
	}
}
```

When you don’t define an access modifier the class and members are by default public

Access modifiers

- The 3 access modifiers are
  - public
  - private
  - protected

- Protected is discussed with inheritance in chapter 8.

Demo1

Static

- Static members are the same for all instances.
- Static methods:
  - Call only other static methods
  - Access only static data
  - Do not have access to this.
- Static blocks:
  - Can be used to initialise a class.

Demo 2 & Demo 3

Overloading

- Multiple methods can share the same name as long as their parameter declarations are different.

```
int add(int a, int b){
}

double add(double a, double b){
}
```

varargs

- Another method for flexible method parameters is variable length arguments.
- In this case v is an array of int's

```
public static int add(int ... v){

}
// This is how the function could be used
add(1,54,7,8,9,376);
```

varargs

```
public static int add(boolean b, int ... v){
    //This is OK

  }

public static int add(int ... v, boolean b){
      //This is not OK.
      //The varargs declaration must always be at the the end.
  }

public static int add(int ... v, double ... d){
      //This is not OK.
      //Only one varargs is permitted per method.
  }

public static int add(int a, int ... v){
      //This is not OK.
  }
```

Demo 4

Overloading constructors

```java
public class Demo5 {
    private double accountBalance;

    Demo5(){
        this.accountBalance = 0;
    }

    Demo5(double startingBalance){
        this.accountBalance = startingBalance;
    }
}
```

Demo 5

ARCTIC TIGER

Objects as parameters

- Passed by-reference
- Whereas primitives are always by-value

Demo 6

Nested /inner classes

ARCTIC TIGER

Demo 7

ARCTIC TIGER