Chap06Yourname.java

Value methods

```
import _____;

public class Chap06 {

    _____

    public static void main (String[] arg) {

        _____

        //All the methods we have written so far have been void which do NOT
        //return values. In Chap06, we'll write methods that return values
        //which we call value methods.

        //Compared to void methods, value methods differ in two ways:
        //They declare the type of the return value (the return type);
        //They use at least one return statement to provide a return value. And
        //the type of the expression in the return statement must match the
        //return type of the method.
}
```

```
import java.util.Scanner;

public class Chap06 {
	private static Scanner in;

	public static void main (String[] arg) {
		in = new Scanner(System.in);

			//All the methods we have written so far have been void which do NOT
			//return values. In Chap06, we'll write methods that return values
			//which we call value methods.

			//Compared to void methods, value methods differ in two ways:
			//They declare the type of the return value (the return type);
			//They use at least one return statement to provide a return value. And
			//the type of the expression in the return statement must match the
			//return type of the method.
}
```

6.1 Return values

```
System.out.println("6.1 Return values");
System.out.print("Let me find the absolute value of x = ");
double x = in.nextDouble();
System.out.printf("OK, it's %f.\n", absoluteValue(x));
System.out.println();
```

6.1 Return values

```
public static double absoluteValue(double x) {
        if (x >= 0) {
                return x;
        } else {
                return - x;
        }
        //The return value from this method is a double.
        //Now we have a form of return statement that include a return value.
        //Make sure that every possible path through the program reaches a return
        //statement.
        //Code that appears after a return statement, or any place else where
        //it can never be executed, is called dead code.
}
```

6.2 Incremental development

```
System.out.println("6.2 Incremental development");
//Incremental development starts with a working program and make small,
//incremental changes.

System.out.println("Give me the coordinates of two points, the center"
                   + "of the circle and a point on the perimeter. Then I will"
                   + "tell you the area of the circle");
```

6.2 Incremental development

```
System.out.print("The x-coordinate of the center is x1 = ");
double x1 = in.nextDouble();
System.out.print("The y-coordinate of the center is y1 = ");
double y1 = in.nextDouble();
System.out.print("The x-coordinate of the parameter point is x2 = ");
double x2 = in.nextDouble();
System.out.print("The y-coordinate of the parameter point is y2 = ");
double y2 = in.nextDouble();
System.out.println(circleArea(x1, y1, x2, y2));
System.out.println();
```

6.2 Incremental development

Step 1:

```
        public static double distance
                (double x1, double x2, double y1, double y2) {
                return 0;
}
//A stub is an outline including the method signature and a return
//statement. The return statement is a placeholder for the program to
//compile.
```

6.2 Incremental development

Step 2:

```
public static double distance
        (double x1, double x2, double y1, double y2) {
        double dx = x2 – x1;
        double dy = y2 – y1;
        System.out.println("dx is  " + dx);
        System.out.println("dy is  " + dy);
        return 0;
    }
```

//Scaffoldings are codes which are helpful for building the program but not part of the
// final product.

6.2 Incremental development

```
public static double circleArea(double x1, double y1, double x2, double y2) {
        double dx = x2 - x1;
        double dy = y2 - y1;

        _____________________________________

        _____________________________________
        return circleArea;
}
```

6.2 Incremental development

```
public static double circleArea(double x1, double y1, double x2, double y2) {
        double dx = x2 - x1;
        double dy = y2 - y1;
        double radius = Math.sqrt(dx * dx + dy * dy);
        double circleArea = Math.PI * radius * radius;
        return circleArea;
}
```

6.5 Boolean methods

```
System.out.println("6.5 Boolean methods");
System.out.print("Give me a number, and I will tell you whether it"
                    + " is positve or not. x = ");
x = in.nextDouble();
System.out.println(isPosOrNot(x));
System.out.println();
```

6.5 Boolean methods

```
public static boolean isPosOrNot(double x) {

_____

}
```

6.5 Boolean methods

```
public static boolean isPosOrNot(double x) {
        return x > 0;
}
```

6.7 Factorial

```
System.out.println("6.7 Factorial");
System.out.print("Give me a non-negative integer, and I will tell "
                    + "you its factorial. n = ");
int n = in.nextInt();
System.out.printf("n factorial is %d.\n", factorial(n));
System.out.println();
```

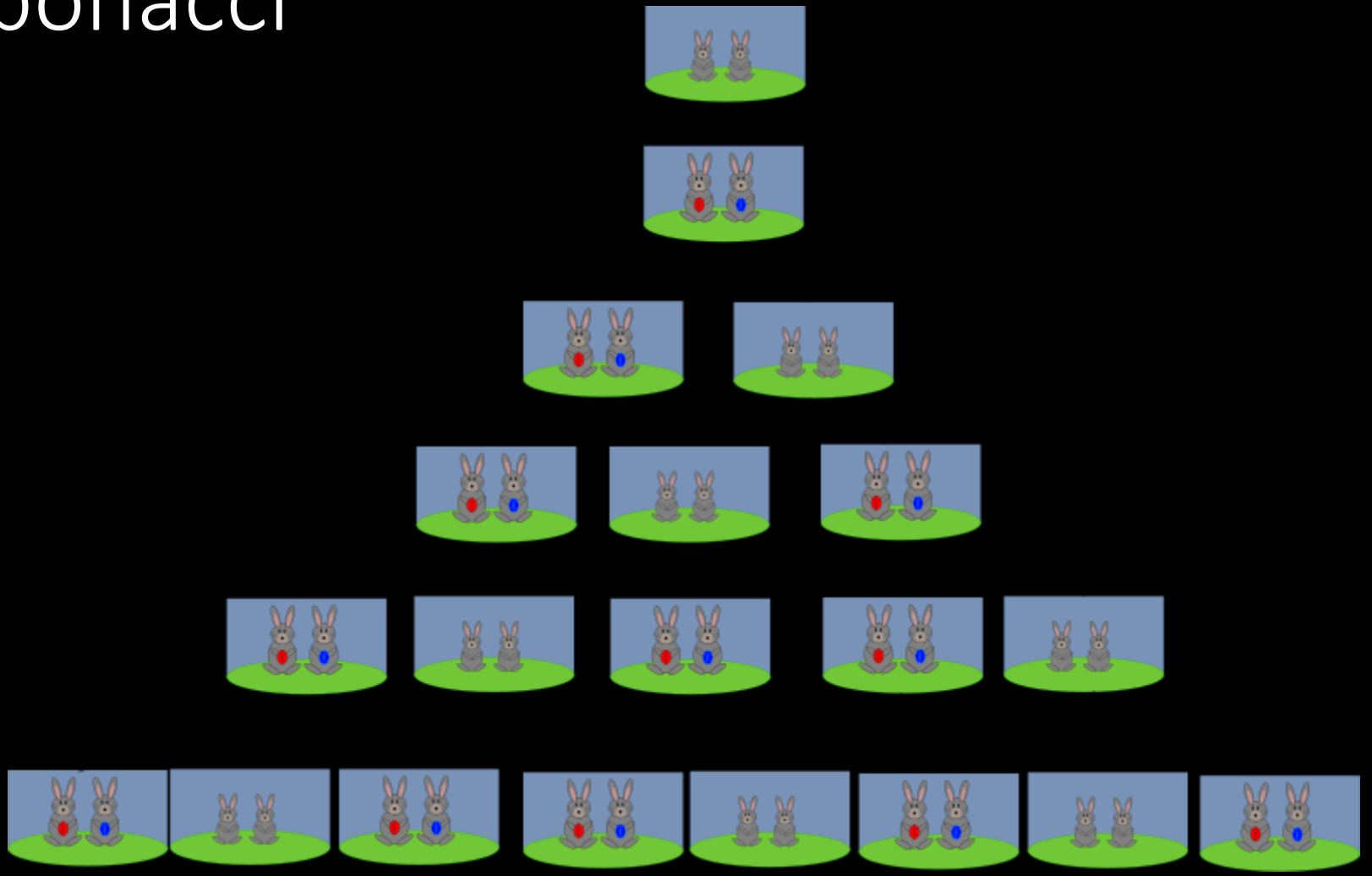
6.7 Factorial

```
public static int factorial(int n) {
_____________________
_____________________________________________
_____________
_____________________________
_____________________________________
_____________________
_____________________________________________________________________________
_____________________________________
_____________________
_____________
}
```

6.7 Factorial

```
public static int factorial(int n) {
        if (n > 0) {
                return n * factorial(n - 1);
        } else {
                if (n == 0) {
                        return 1;
                } else {
                        System.out.println("n must be NON-NEGATIVE.");
                        return -1;
                }
        }
}
```

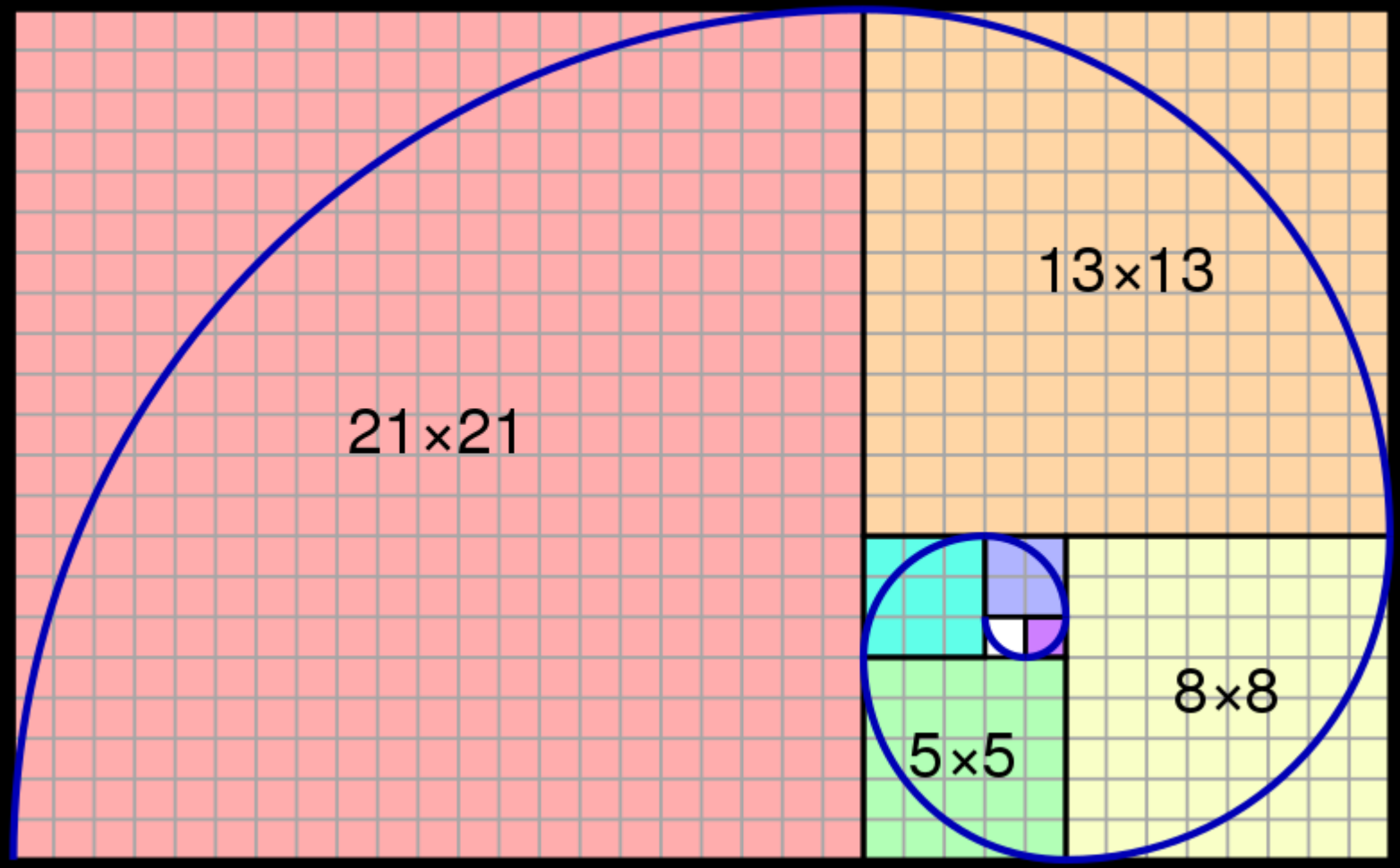
6.7 Fibonacci



6.7 Fibonacci

$$F_1 = 1$$
$$F_2 = 1$$
$$F_n = F_{n-1} + F_{n-2}$$

6.7 Fibonacci



Johannes Kepler observed that the ratio of consecutive Fibonacci numbers converges.

He concluded that the limit approaches the golden ratio.

6.7 Fibonacci

```
System.out.println("6.9 Fibonacci sequence");
System.out.print("We want to find the ith number in the Fibonacci "
                    + "sequence. n = ");
int i = in.nextInt();
System.out.printf("That will be %d.", fibonacci(i));
```

6.7 Fibonacci

```
public static int fibonacci(int i) {
_____
_____
_____
_____
_____
}
```

6.7 Fibonacci

```
public static int fibonacci(int i) {
        if(i <= 2) {
                return 1;
        } else {
                return fibonacci(i - 2) + fibonacci(i - 1) ;
        }
}
```