


**Top-Down Programming**

Lewis D Griffin

Department of Computer Science  


1

---

---

---

---

---


---

---

---

**Lecture Contents**

- top-down programming
- static methods
- object references
- cohesive methods

Department of Computer Science  


2

---

---

---

---

---


---

---

---

**What is Top-Down Programming?**

- A code-centric approach to programming.
- Uses sequences and nested levels of commands
- Usage peaked in the late 70's and early 80's.
- Replaced by data-centric approaches.

Department of Computer Science  


3

---

---

---

---

---

---

---

---

## Using Methods

- Instead of writing one long list of instructions and executing from start to finish.
- We can separate out the instructions into different methods.
- And call the methods one after another.
- Immediately improves code readability.

4

---

---

---

---

---

---

---

---

## Step 1 towards top-down programming

```
main(){  
  S1;  
  S2;  
  S3;  
  S4;  
  S5;  
  S6;  
}
```

BAD STYLE

```
main(){  
  M1();  
  M2();  
}  
  
M1(){  
  S1;  
  S2;  
  S3;  
}  
  
M2(){  
  S4;  
  S5;  
  S6;  
}
```

BETTER STYLE

5

---

---

---

---

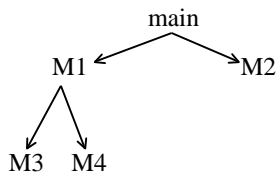
---

---

---

---

## Step 2 towards top-down programming



### Execution order

- main calls M1
- M1 calls M3
- M1 calls M4
- main calls M2

6

---

---

---

---

---

---

---

---

## Top-down is a Programming Style

- This style of programming is known by several names:
  - Top-down programming
  - Procedural decomposition
  - Structured programming
  - and others

7

---

---

---

---

---

---

---

---

## Top-down is an approach to problem solving, through recursive decomposition

- Consider a problem to be solved:
  - If it is simple  
    solve it directly,
  - if it is complex  
    break it into sub-problems  
    & solve each in turn.
- Also apply this procedure to each sub-problem.

8

---

---

---

---

---

---

---

---

## The *static* keyword (first look)

- In fully procedural programming all variables and methods are static (so one doesn't notice the concept).
- In object-oriented programming most variables and methods are non-static (instances).
- Whenever non-static code is executed it is always in the context of some piece of data in memory.
- Static code has the entire memory as context.

9

---

---

---

---

---

---

---

---

## Top-down Example

- Write a program to read in a sequence of integers, sort them, and print the sorted sequence.

```
public static void main(String[] args)
{
    //readIntegers;
    //sortIntegers;
    //printIntegers;
}
```

10

---

---

---

---

---

---

---

---

## readIntegers

- Input?
- Output?
  
- Lets store the integers in an array.
- Use a loop to read them in.

11

---

---

---

---

---

---

---

---

## readIntegers

```
/**
 * Reads in 100 integers.
 */
public static int[ ] readIntegers()
{
    int[ ] vals = new int[100];
    // Loop to read in integers
    return vals ;
}
```

12

---

---

---

---

---

---

---

---

## sortIntegers

- Input?
- Output?

```
/**
 * Sorts an array of integers.
 */
public static int[] sortIntegers(int[] vals)
{
    // Do the sorting
    return vals;
}
```

13

---

---

---

---

---

---

---

---

## printIntegers

- Input?
- Output?

```
public static void printIntegers(int[] vals)
{
    // loop to print integers
}
```

14

---

---

---

---

---

---

---

---

## Amend main method:

```
public static void main(String[] args)
{
    int[] vals = readIntegers();
    vals = sortIntegers(vals);
    printIntegers(vals);
}
```

15

---

---

---

---

---

---

---

---

### Ways of gradually filling in programs when doing top-down.

- `// read in integers`
- `// TODO – read in integers`
- `readInIntegers(vals);`
- `static void readInIntegers(int[] vals){/* TODO – write code */}`
- ```
static void readInIntegers(int[] vals){
    // TODO – replace this dummy code
    for(int i:<vals.length;i++)
        vals[i]=i;
}
```
- Actually write the function!

16

---

---

---

---

---

---

---

---

### A variant using object references

```
public static void main(String[] args)
{
    int[] myInts = new int[100];
    readIntegers(myInts);
    sortIntegers(myInts);
    printIntegers(myInts);
}
```

- Q: What's the difference in how these methods deal with arguments and returning data?

A: use of an *object reference* in calls to read and sort.

17

---

---

---

---

---

---

---

---

### Why are these methods better...

- In original version, `readIntegers` created and array and read data into it.
- In the improved version it only does *one* thing (reads in data).
- A method that does only one thing (and does it well) is said to be *cohesive*.
- Split up your methods into cohesive sub-methods.

18

---

---

---

---

---

---

---

---

## Top-down programming?

- Do we always want to design and write programs in this way?

Yes and No!

19

---

---

---

---

---

---

---

---

## Yes...

- Top-down decomposition is useful for solving small-scale problems,
- and, in some cases, for solving sub-parts of a larger problem.

20

---

---

---

---

---

---

---

---

## No...

- Top-down decomposition is a poor way for solving large problems and for designing large programs.
- In fact, it is often a **complete disaster!**
- The method simply does not scale-up.

21

---

---

---

---

---

---

---

---

## A better way

- Object-oriented design and programming provides much better solutions to the construction of larger programs.
- We will examine this technique and its advantages for the remainder of this term.
  
- But also read the text book now.

22

---

---

---

---

---

---

---

---

## For now...

- Top-down solution is rigid, inflexible and fragile.
- OO solution is general purpose, flexible and robust.

23

---

---

---

---

---

---

---

---

## Summary

- Top-down programming is one way of solving problems.
- It is useful for small-scale problems or sub-problems.
- It does not scale-up.

24

---

---

---

---

---

---

---

---