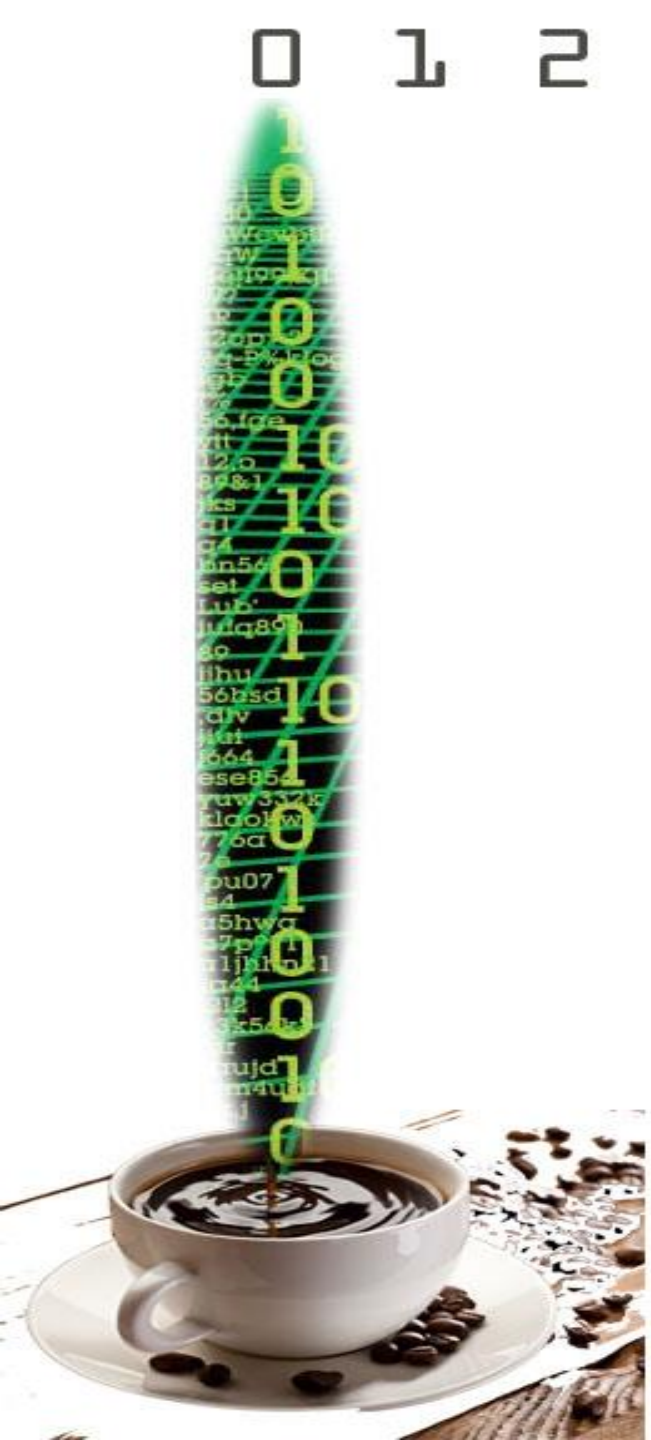


Object–Oriented Programming

Data Abstraction in C++

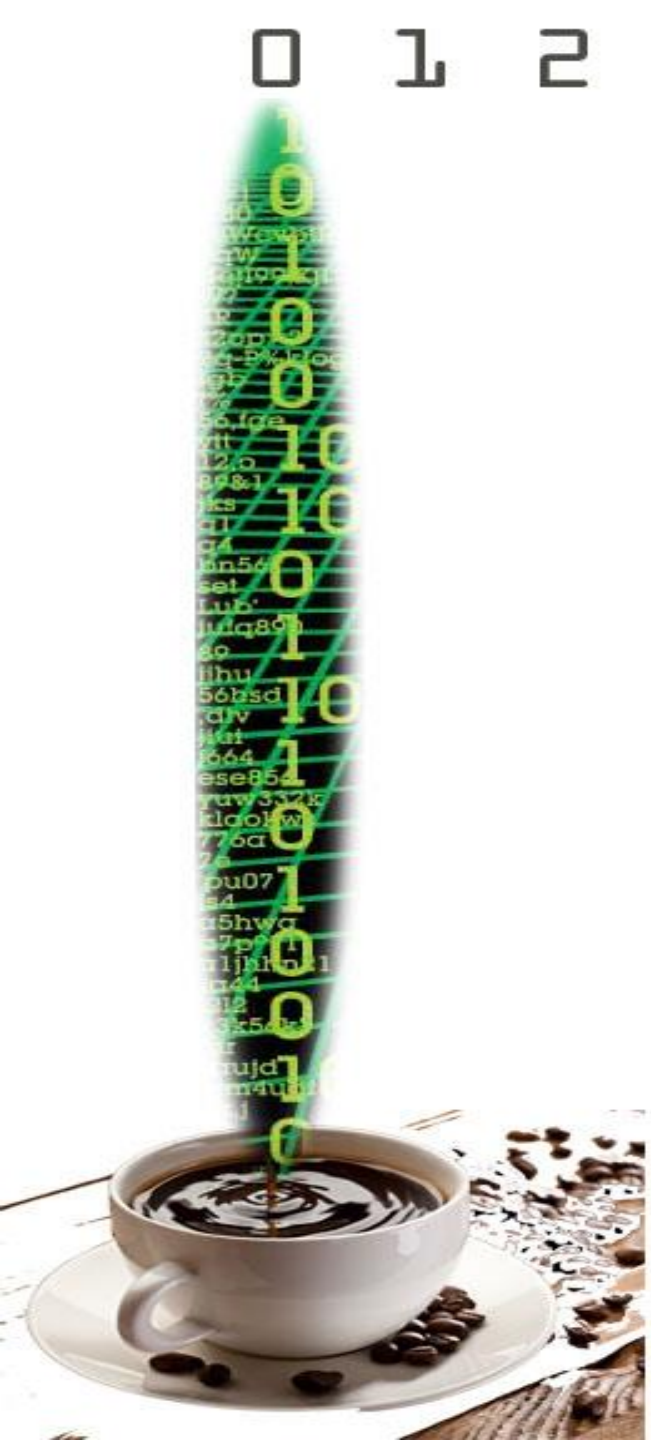
Dimitrios N. Terzopoulos

terzopod@math.auth.gr

21/10/2015

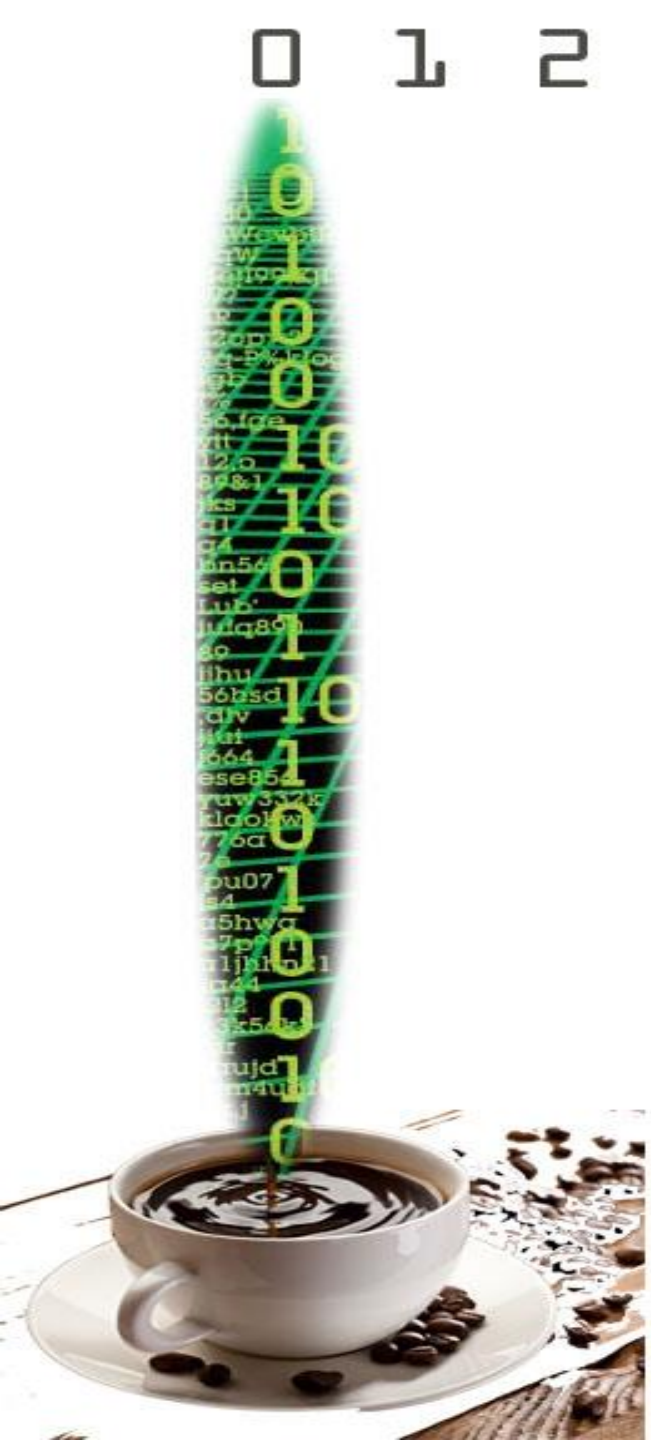
Curriculum

- Data Abstraction
- Client/Library Development (Design)
- Encapsulation – Protection – Scoping
- Code Readability – Coding Standard
- Object Oriented Programming



Practice in C++

- Defining Classes
- Access Specifiers
- Initialization – Cleanup (Construction – Destruction)
- Namespaces – Scope Resolution
- Member Functions – Operators – Overloading



Data Abstraction

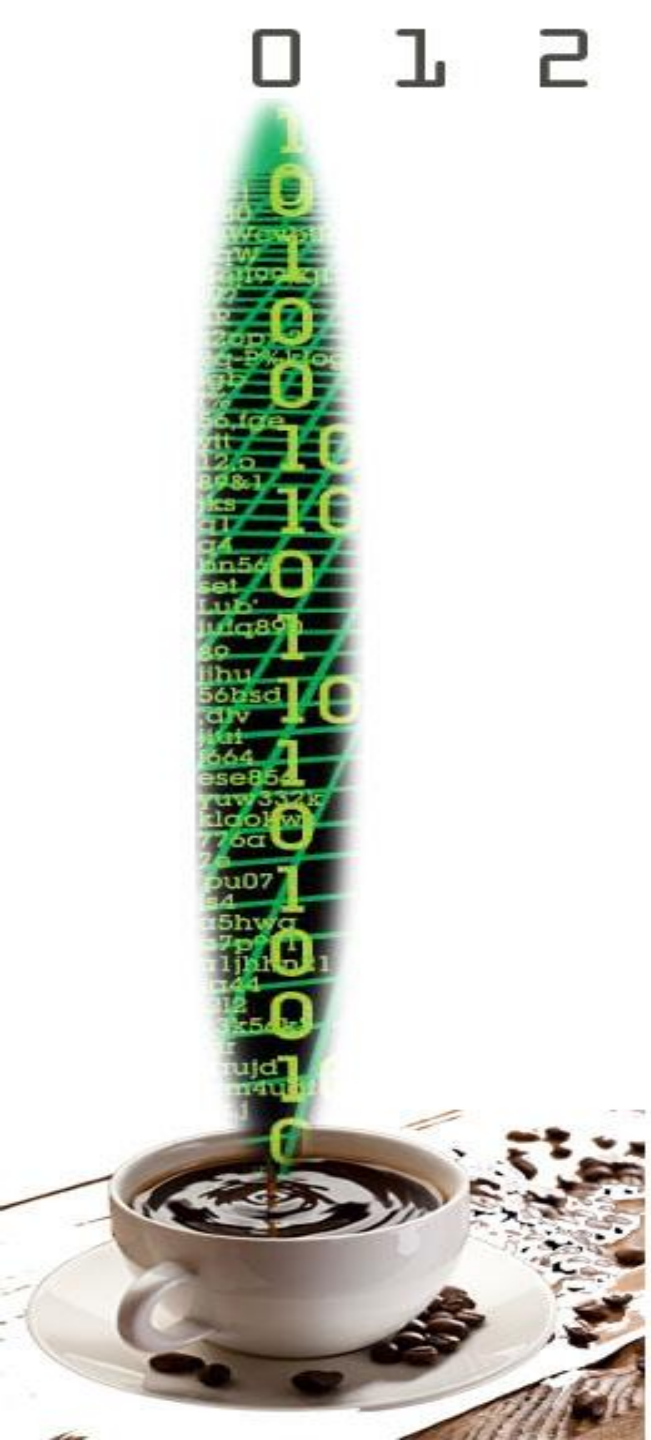
- **Level**: *Lower* (Bits) to *Higher* (Real World).
- **Abstraction**: Scaling of Complexity.

Hiding Implementation Details

Bits $<$ Bytes $<$ Numbers $<$ Dimensions $<$ Elements

(Example)

Properties $<$ Behavior $<$ Interactions $<$ Real–World!!!



Data Abstraction

What do you know about...

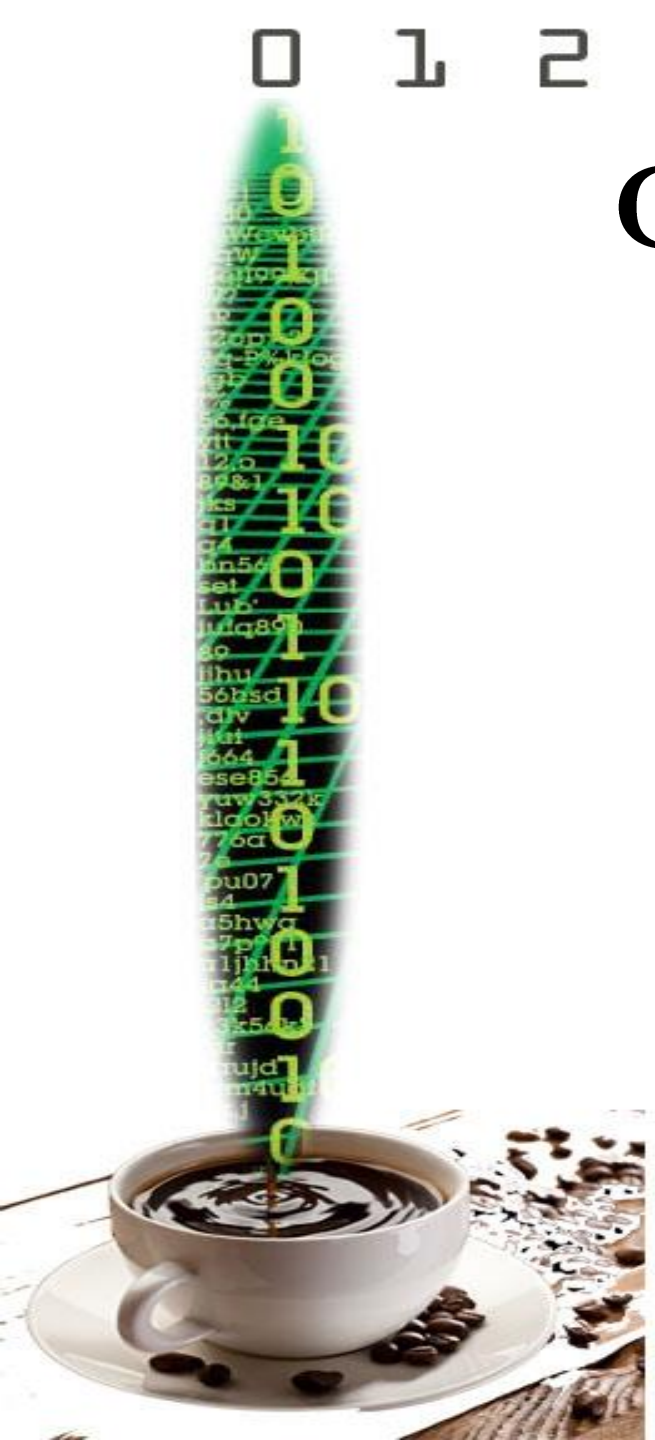
- Electronic Circuits (Transistors, Capacitors, Diodes...)

  But you use Cameras, Cellphones, Computers, TVs
- Chemistry (Elements, Ions, Reactions, Bonds...)

  But you use Detergents, Softeners, Conditioners, Medicine

- *Numbers, Algebraic Structures*

We hide the intricate details!



Client / Library Development

- **Libraries**: Building blocks of *software*.

- Library Developer: Builds ...libraries(!)
- Client Developer: Combines libraries into a program.

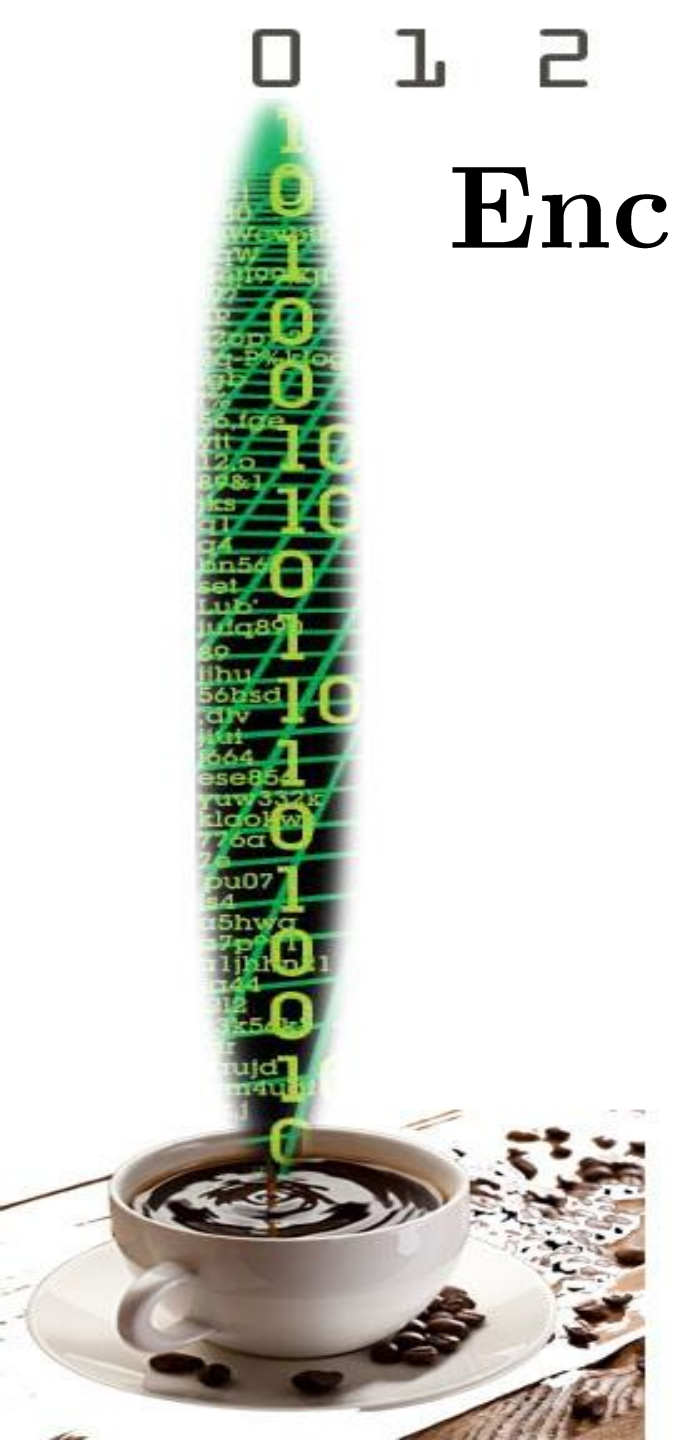
Libraries

- Geometric Elements, Shapes
- Shading, Rendering, Lighting
- Physics, Collisions, Ray-Tracing,

➔

Final Program

Computer Game



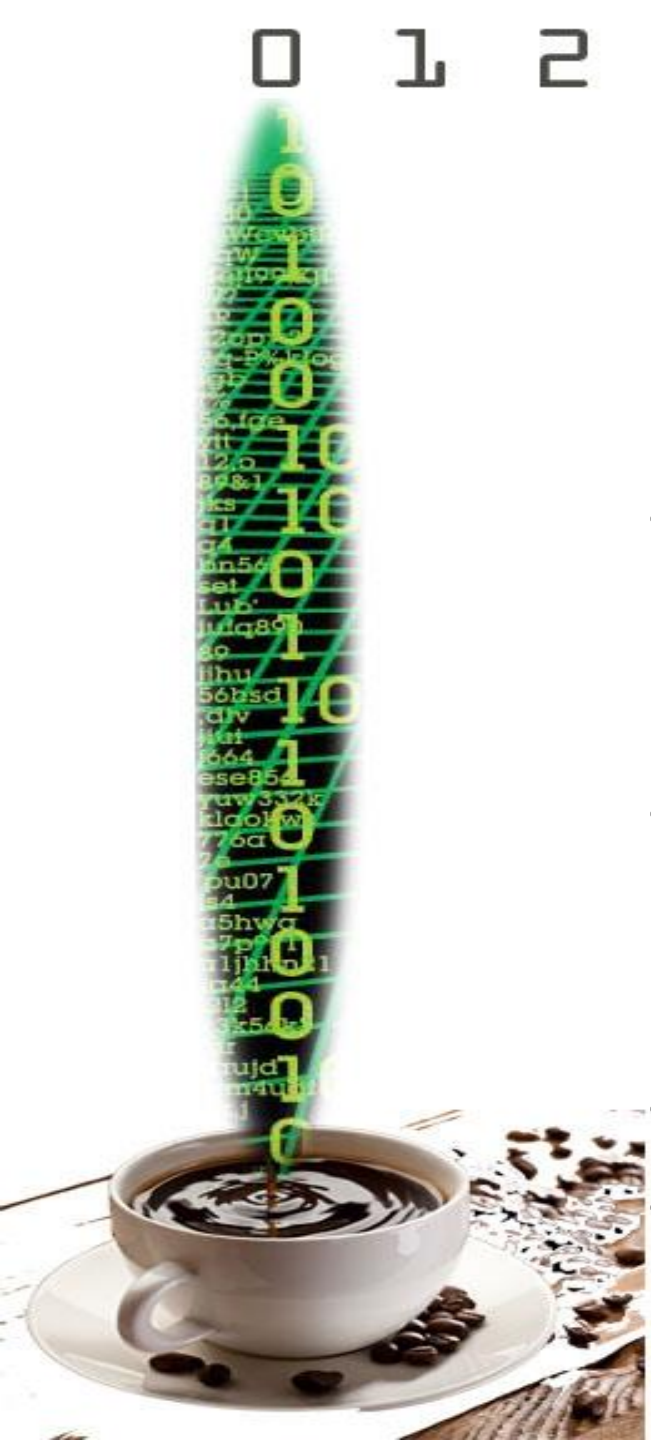
Encapsulation–Protection–Scoping

- Encapsulation: Put *functions* and *data* together.
- Encapsulation: Restrict *access* to the data.

Distinct meanings → Both important in OOP

- Every object has its ***secrets***.

- Scoping: *Who* sees *what*, from *where*.
  The designation of *environments* – “Code workplaces”.



Examples

Encapsulation (I)

- Circle {Radius, Center, *getArea*(), *getLength*()}

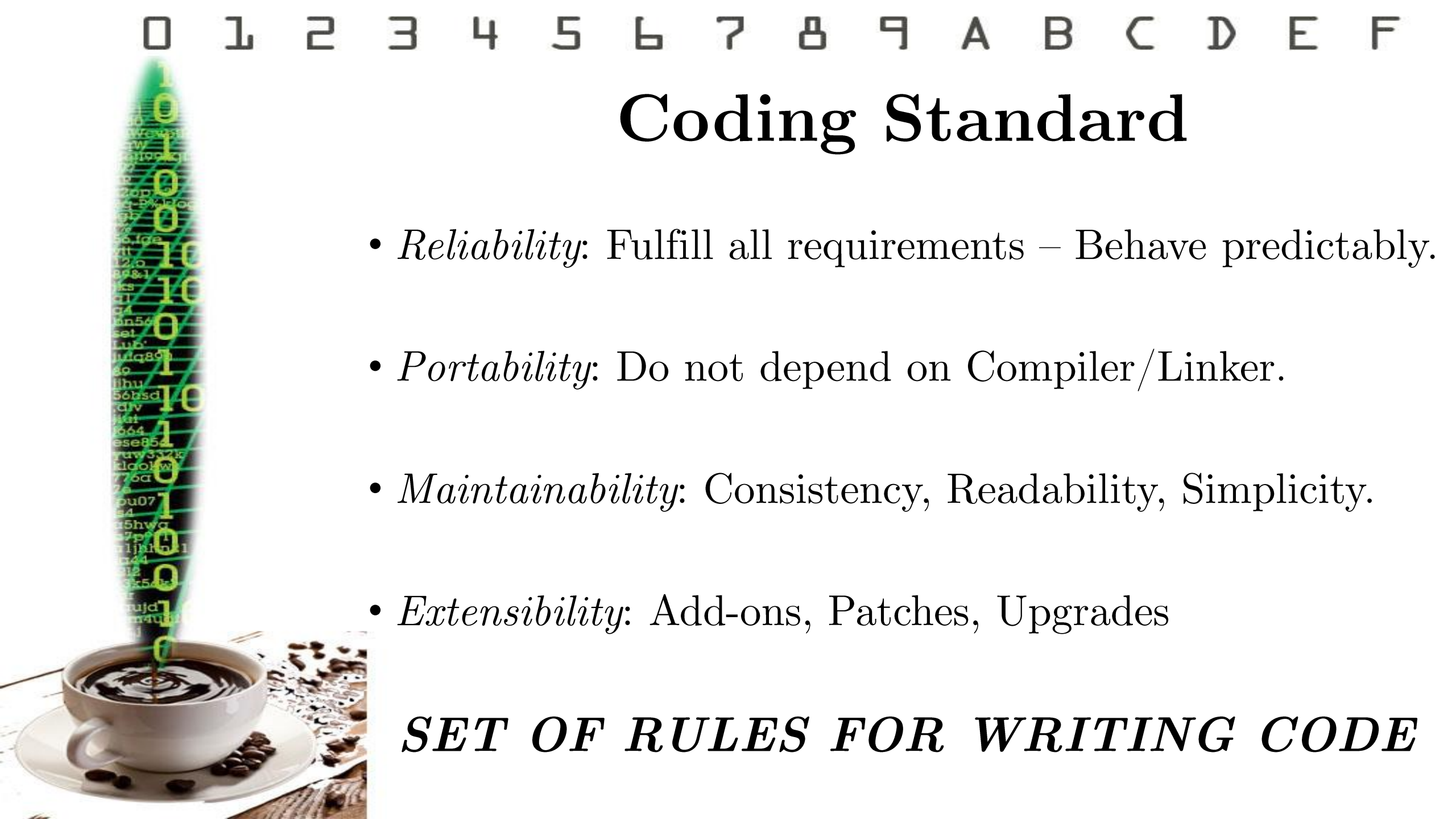
Encapsulation (II) – **Member Protection**

- Circle {*private*: Radius, Center | *public*: getRadius()}

Variable Scoping

- Namespace "*MyPlace*" {Helen, Mike, ***George***, Thomas}
- Namespace "*YourPlace*" {Timothy, ***George***, Jennifer}

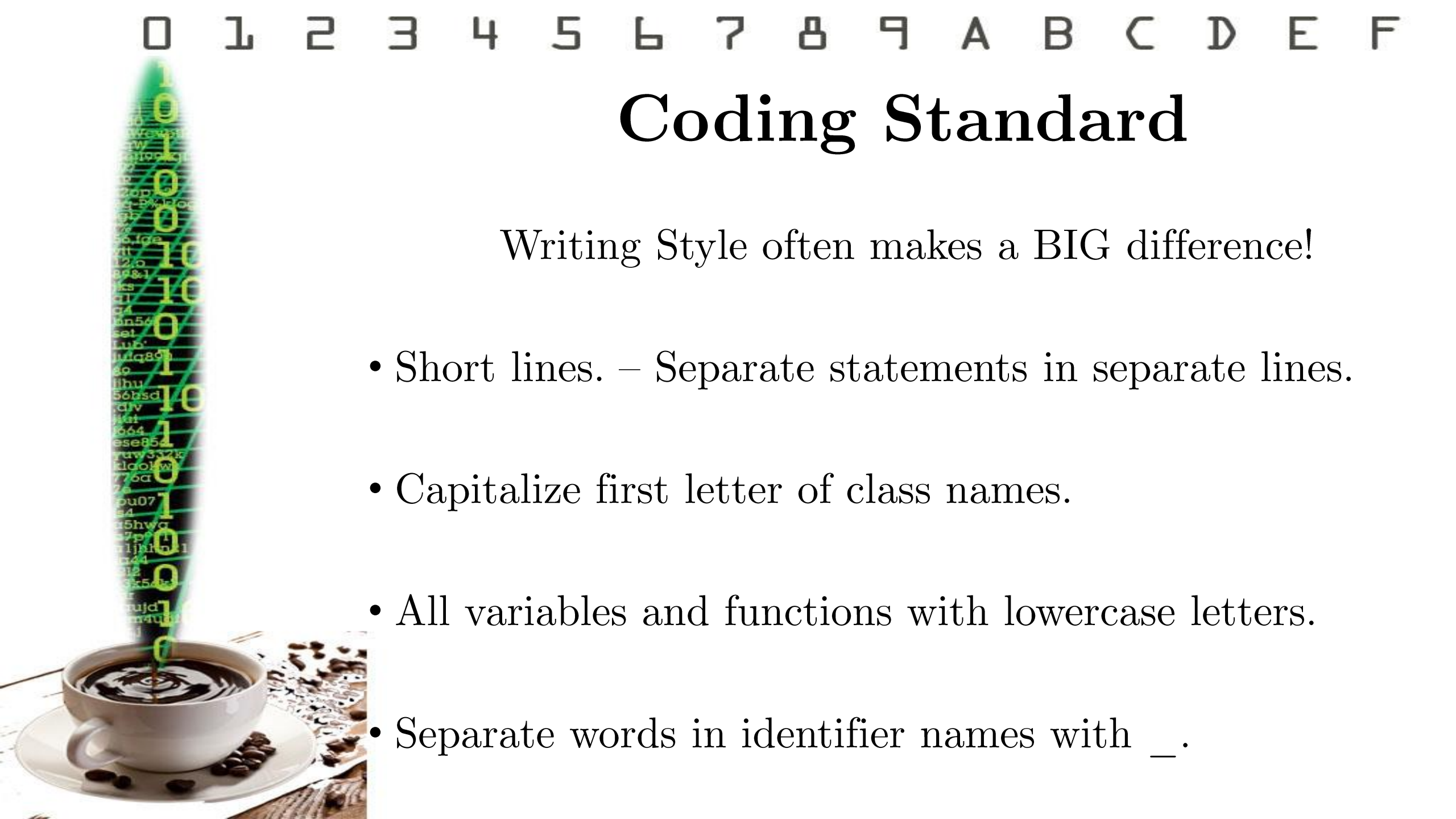
George who? → *MyPlace*::***George***



Coding Standard

- *Reliability*: Fulfill all requirements – Behave predictably.
- *Portability*: Do not depend on Compiler/Linker.
- *Maintainability*: Consistency, Readability, Simplicity.
- *Extensibility*: Add-ons, Patches, Upgrades

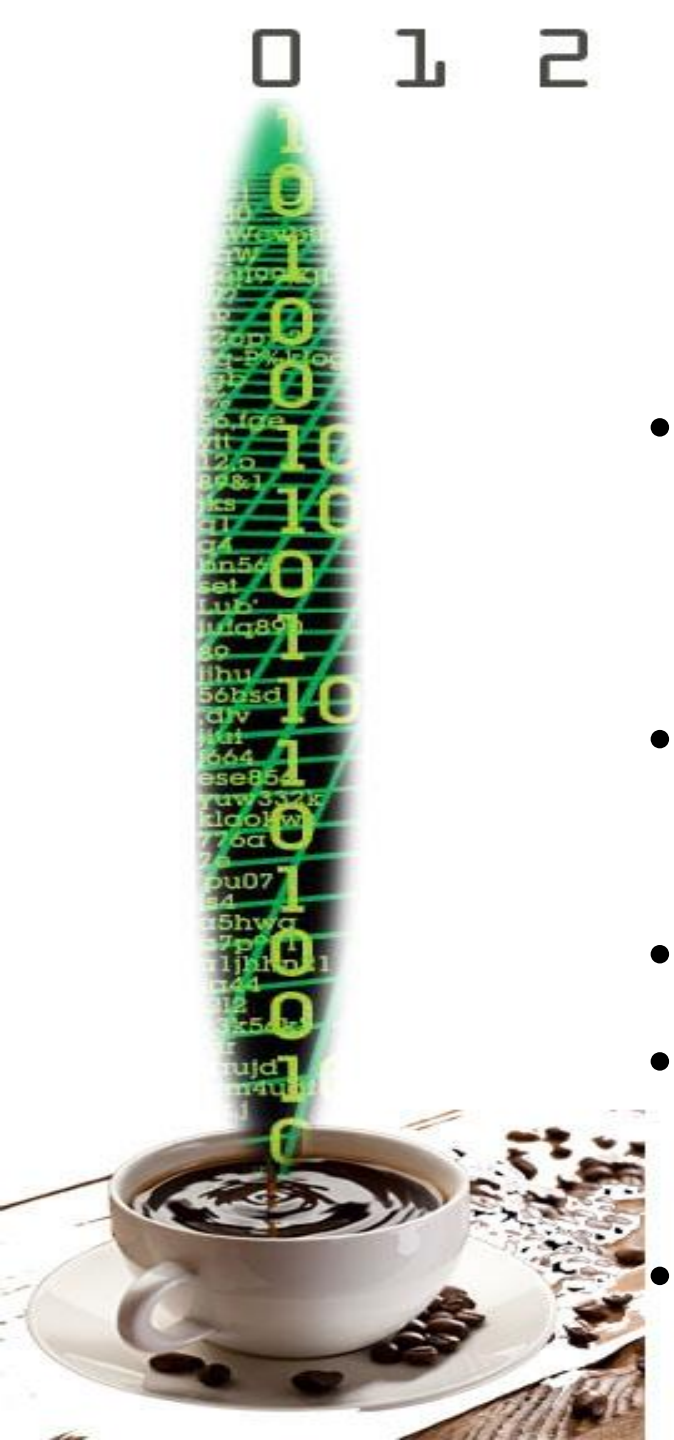
SET OF RULES FOR WRITING CODE



Coding Standard

Writing Style often makes a BIG difference!

- Short lines. – Separate statements in separate lines.
- Capitalize first letter of class names.
- All variables and functions with lowercase letters.
- Separate words in identifier names with _.



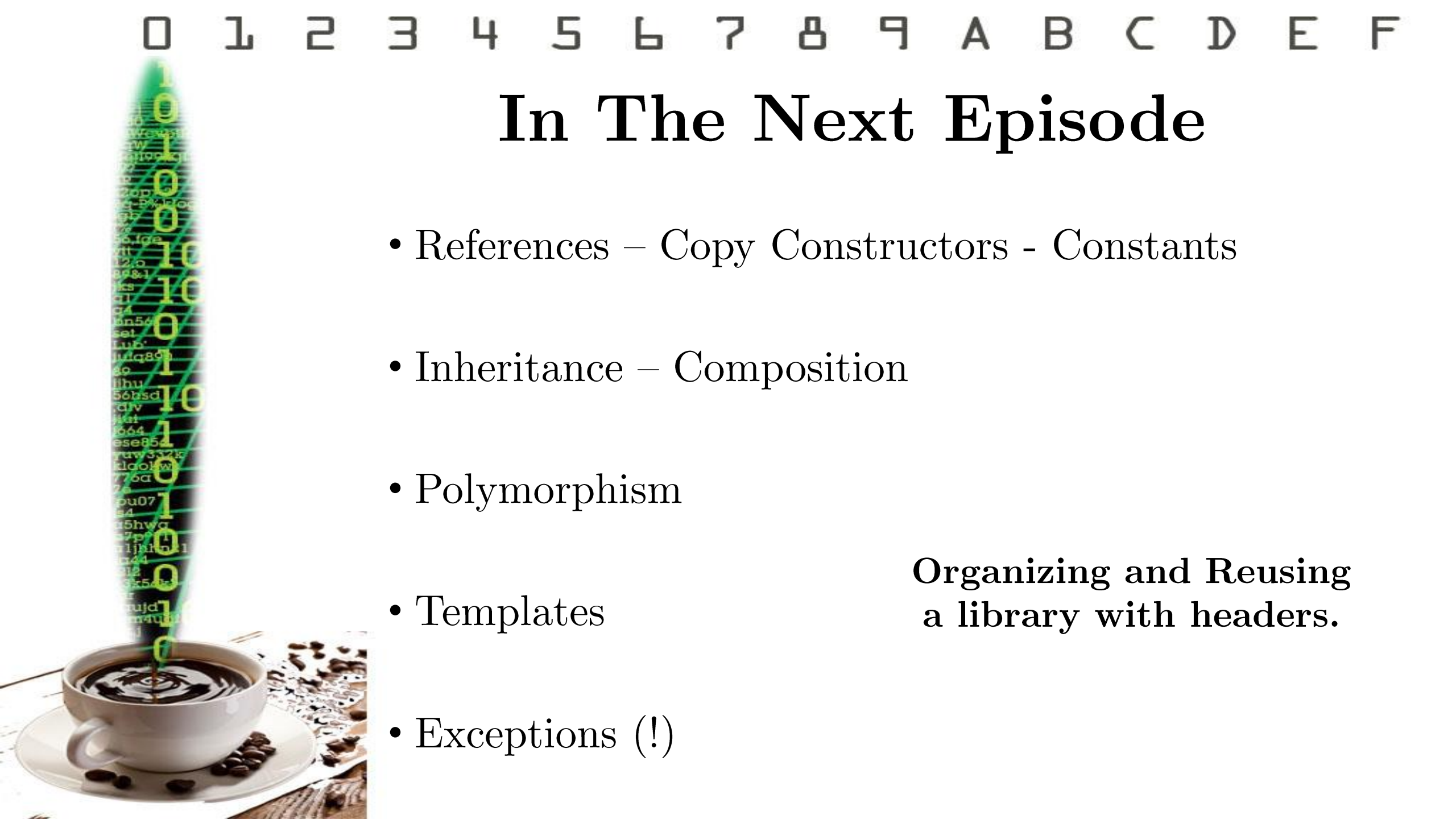
Object Oriented Programming

- Classes consist of **data** and **behavior** – Like objects.
  (i.e. **variables** and **functions**)

- Classes limit access to these variables and functions.

- Class definition → Prototype
- Class implementation → Instance

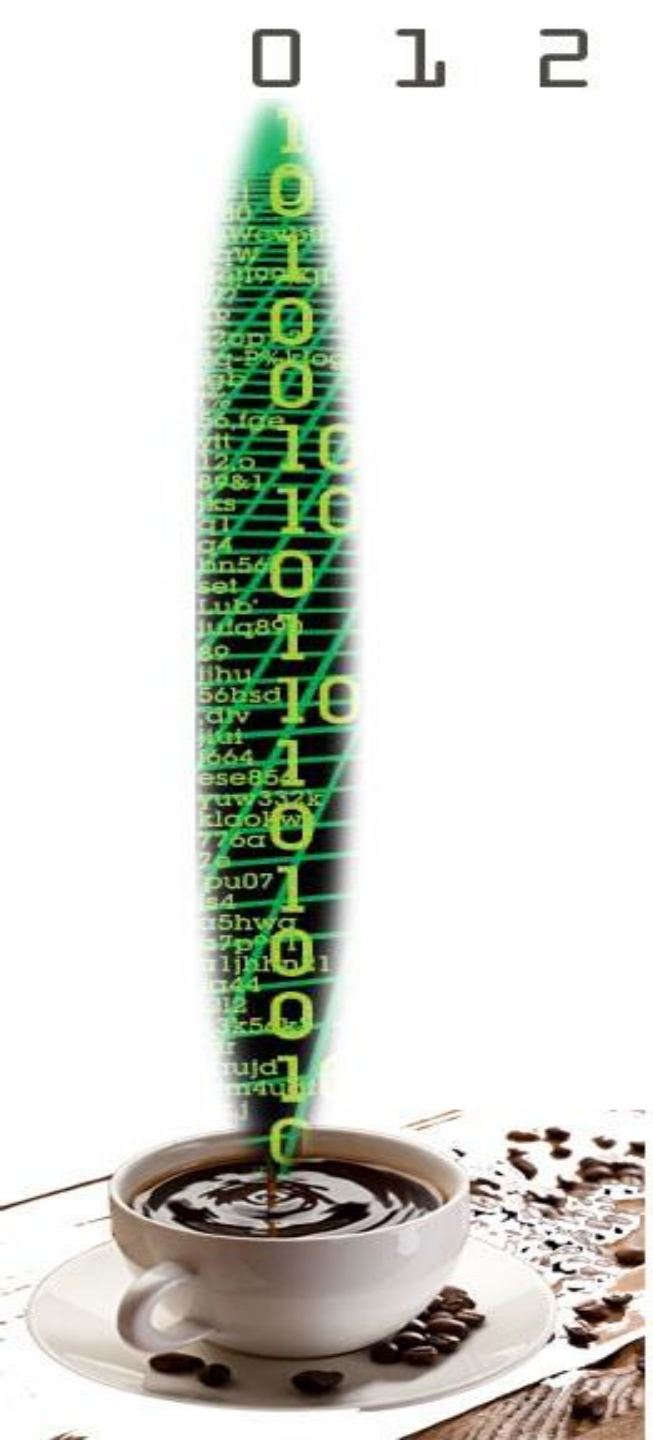
- Classes "work" together to simulate a "code factory".



In The Next Episode

- References – Copy Constructors - Constants
- Inheritance – Composition
- Polymorphism
- Templates
- Exceptions (!)

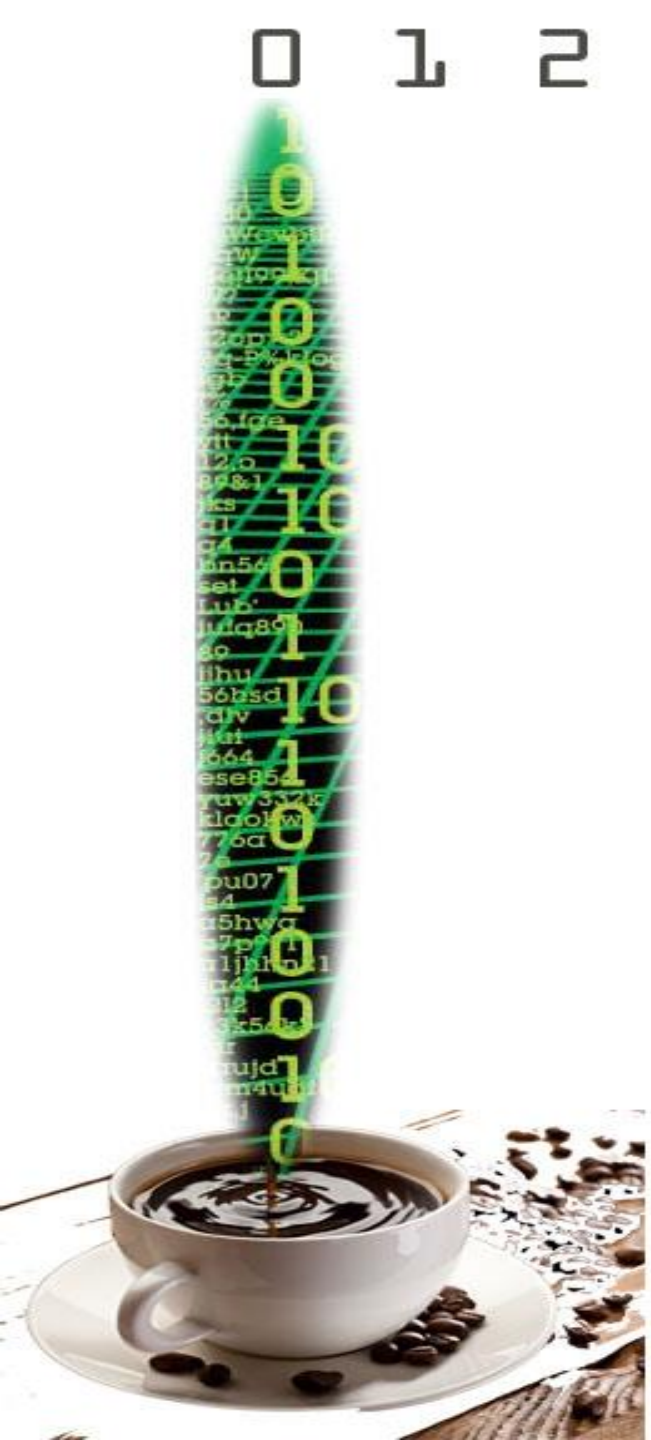
Organizing and Reusing a library with headers.



Building a Small Library

- **In This Episode**: Elementary Dynamic Array

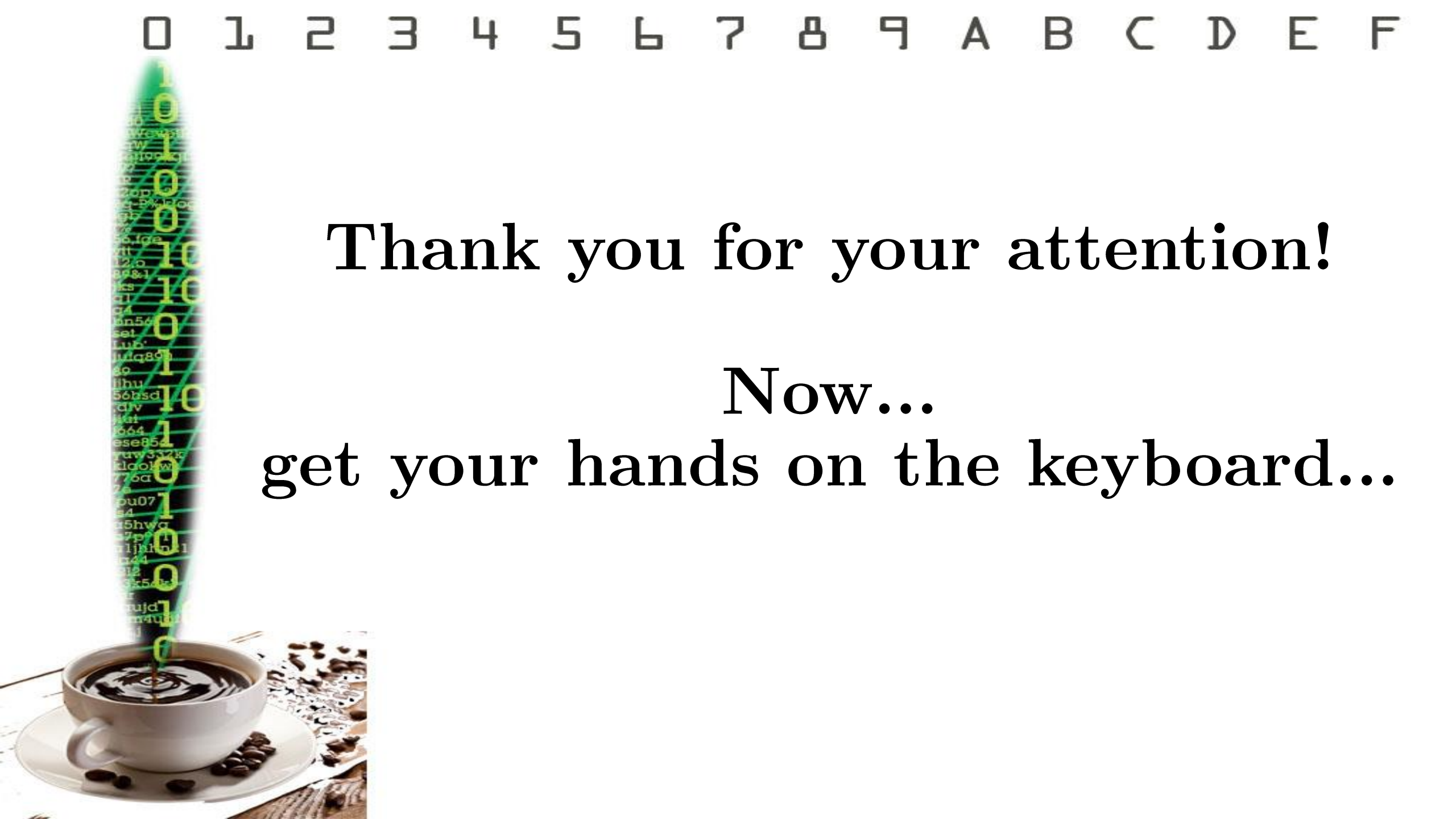
- **In The Next Episode**: Pick your poison...
  - Stack / Heap
  - Small Complex Number Library
  - Doubly Linked List
  - Disjoint Set (For Graph Implementations)
  - CRS (Compressed Row Storage) Sparse Matrix
  - Elementary Geometric Library (CAD Applications)
  - Or...



References

- Stroustrup, B. (2013). *The C++ Programming Language – Fourthe Edition.* Addison–Wesley, Pearson Education, Oxford.

- Eckel, B. (2000). *Thinking in C++* (2 Volumes). Prentice Hall, Pearson Higher Education, New Jersey.



Thank you for your attention!

Now...
get your hands on the keyboard...