

CSC110 Spring 2016

Class 1: Introduction

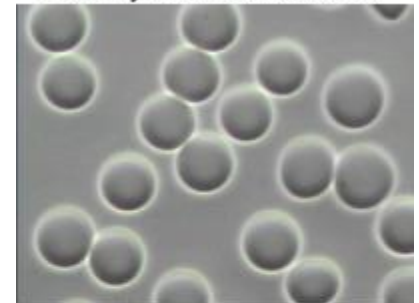
Instructor:
Dr. Tibor van Rooij

Acknowledgement: Thanks to Drs. Zastre, Berg, Jackson, and Tory who have made their course materials available to me.

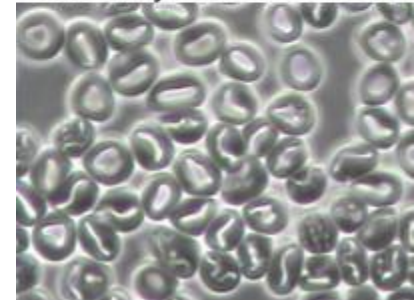
Why Might You Want to Be Here?

- Did you know that these are healthy blood cells (top) and unhealthy (bottom)?
- An expert may need only a few minutes to determine this for one sample
- However when you multiply those minutes by an increasing number of people waiting to be screened
- You may run out of expert lab technicians to perform all the checks in the time available.

Healthy Red Blood Cells



Unhealthy Red Blood Cells

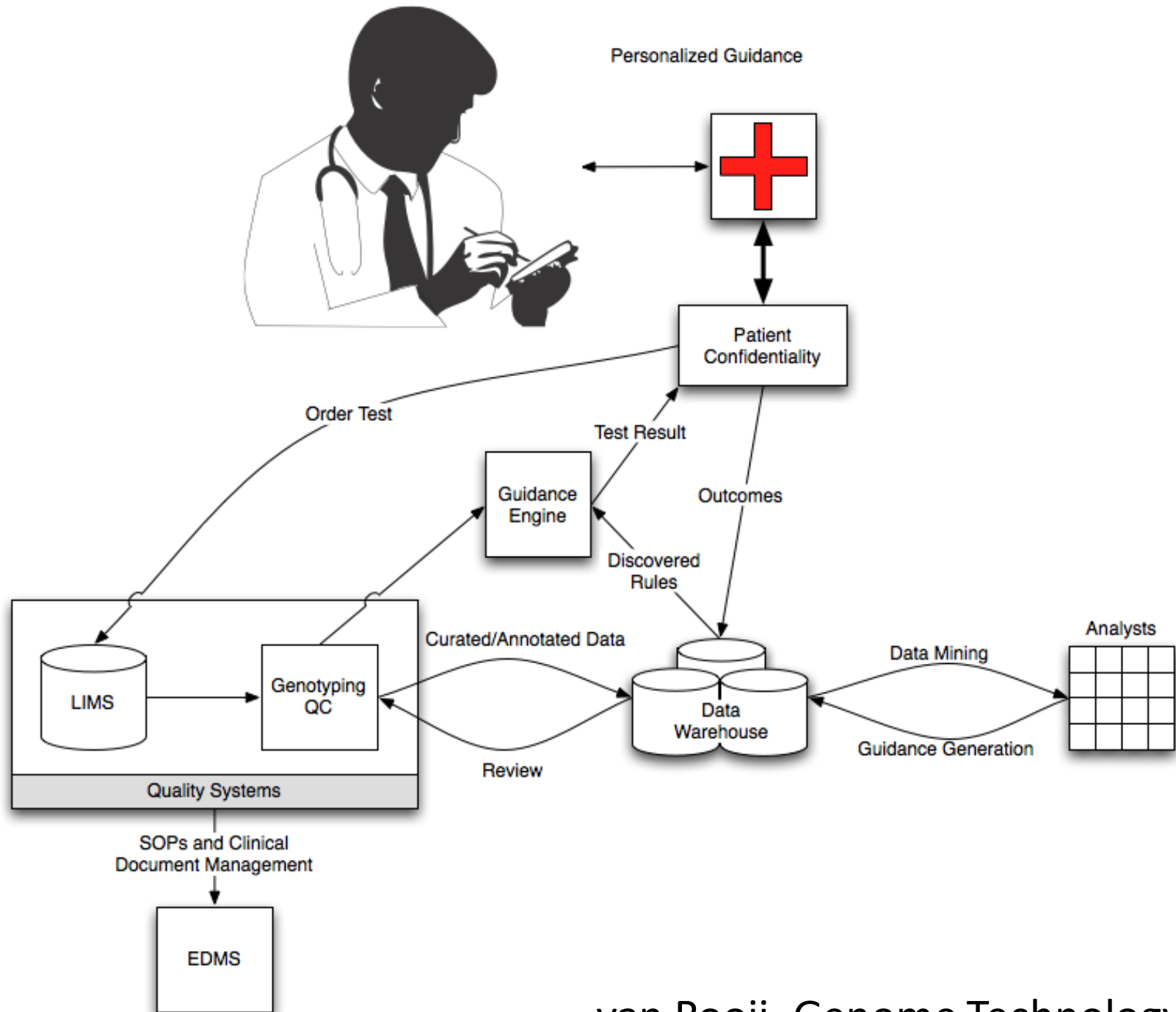


How Can Programming Help?

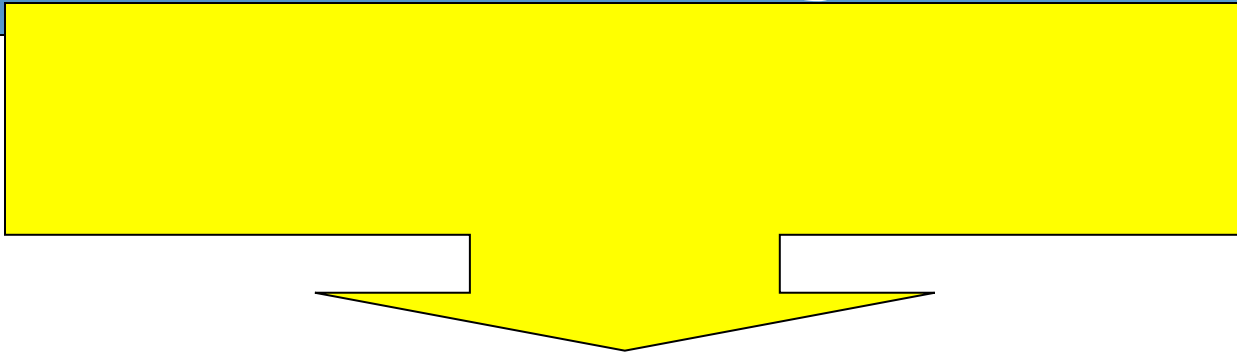
- Programs can be written to:
 - Measure the difference between two pictures
 - Count the number of oddly shaped platelets
 - Analyze gene sequences
 - Analyze the spread of disease
 - Predict the effects of earthquakes
 - Analyze fluctuations in the stock market
 - Track global warming
 - Provide decision-making support
 - Be part of screening programmes for early detection of cancer
 - ... and this list goes on and on, it is virtually endless and bound only by your imagination and creativity :)

In my own practice:

- As part of my previous work:
 - Laboratory Information Management Systems, Genomic data analysis applications, Clinical Decision Support Pipelines (Perl, Java, R)
- As part of my research:
 - Bioinformatics, Medical informatics (Perl, Java, R, Visual Basic, Ruby, C, C++)
 - Medical App Development - iPad, iPhone, Android phone & tablet development (Java, JavaScript, PHP)



CSC 110 Objective



To introduce you to the design and implementation of high-quality software using a programming language.

And to ensure you have a good time and some fun while doing this!

This course is intended to:

- introduce the **methodology** of **computer programming** through the **design and implementation of software** using a **programming language**
- spark interest in the **science of computing** and information processing, and develop an appreciation for its **applicability to a wide range of disciplines**

At the end of this course you will be able to

- design and implement solutions for simple computing and information processing problems
- interpret, analyze, and critique simple computer programs written by others
- learn about intermediate-level programming language features and programming techniques
- continue studying and experimenting with computer programming

Participation

- Lectures:
 - Ask questions, be interactive!
 - Be active in your learning - some class work will be integrated into the lecture (not for marks)

- Two hours per week you will have a hands-on lab:
 - In the computer lab

- Assignments:
 - **Are to be completed independently.**
 - Many resources are available to you that will help when completing assignments.

CSC 110 Overview

- | | |
|---|--|
| <ul style="list-style-type: none">• Computer organization• Syntax & semantics of a high-level language (Java)• Variables, expressions, and assignment• Representation of numbers• Input / Output• Conditional and iterative control structures• Functions/methods and parameter passing• Structured decomposition• Problem-solving strategies | <ul style="list-style-type: none">• The role of algorithms in problem-solving• Implementation strategies for algorithms• Debugging strategies• Arrays• Strings and string processing• Data representation in memory• Linear search• Selection sort• Encapsulation & information hiding• Separation of behavior and implementation• Classes |
|---|--|

Resources

- Textbook: “Building Java Programs”, by Reges & Stepp (third edition), it is a **required textbook** for this course
- conneX Site: (Schedule, Notes, Assignments, Readings, etc.)
 - You must first enable your CSC account (and we assume you already have a Netlink account)
 - To do this go to: accounts.csc.uvic.ca
 - Once the account is enabled, log into connex.csc.uvic.ca
 - If you are enrolled in the course, then you automatically will see a tab on conneX for “csc110”.
- Course outline can be found at:
<https://courses1.csc.uvic.ca/courses/2016/spring/csc/110>

Asking Questions & Finding Help

- **Your text book, and** possibly an alternative Java book
- **Web Sites**
 - Action Item: Find & bookmark some sites (e.g., stackoverflow.com could be useful after the midway point of the semester)
- **Asking questions:**
 - Fellow students, you are free to discuss challenges together!
 - On conneX: Discussion section (chat room)
 - Computer Science Assistance Centre (ECS second floor): hours will be posted
 - Lab Instructors – ask questions during the lab
- **Instructor Office Hours**
 - Tibor van Rooij: Mondays 11:30 to 1:30; Thursday 11:30 to 1:30
email: vanrooij@uvic.ca (please include "CSC 110" in subject line)

Expectations

- Please attend lectures – you are responsible for any course material you miss
- Be considerate to your fellow students:
 - Arrive on time
 - One person talks at a time
 - Be patient if you already understand something – others may be new to the concept

Plagiarism:

- Submitted work may be checked using plagiarism detection software. Cheating, plagiarism and other forms of academic fraud are taken very seriously by both the University and the Department. You should consult the following link:

<http://web.uvic.ca/calendar2008/FACS/UnIn/UARe/PoAcI.html>

for the UVIC policy on academic integrity. Please note that the university policy includes the statement that "**A largely or fully plagiarized assignment should result in a grade of F for the course**".

- Key point:
 - If the work you submit is for personal credit then it must be your **own** work, Please do not hand in other people's work.

Admin for this course

Pre-requisite courses (Math 12 or equivalent):

- If you do not have the pre-requisite course(s) or a waiver for this class then it is advisable to drop yourself from the class.
- If you do not, the UVic student-registration system will drop all such students during the first week and a pre-requisite drop is shown on a student's record.

Taking the course more than twice:

- If you are taking the class for the third time (or more) you must request, in writing, permission from the Chair of the Department and the Dean of the Faculty to be allowed to stay registered in the class (this is a University Rule).
- This letter should be given to Sue Butler, Undergraduate Advisor in ECS 512.
- If you do not have permission, you will be dropped from the class.

Any questions or concerns?

- Contact the Undergraduate Advisor : Sue Butler in ECS 512 (cscadvisor@uvic.ca)

To do list!

- Have you found and logged into conneX and located the CSC 110 site?
 - Do you have a Computer Science account?

If not, go to <http://accounts.csc.uvic.ca> and follow the instructions there.

- Have you reviewed the course outline?
- -it is on conneX and you can also find a link to this at the Computer Science department website www.csc.uvic.ca → Current Students → Undergraduate → Undergraduate Courses)
- Have you obtained a copy of the required textbook and started chapter 1 yet?

Announcements

- Labs start next week (Monday)
 - in ECS 250
 - Please attend your registered lab section
- For this week: Continue reading text up to and including Chapter 2, Section 2
- Assignment 1 will be posted next week – make sure to get started!
- Need extra help help?
 - TA office hours (to be posted)
 - Lab time
 - Computer Science Assistance Centre

What is programming?

Programming is the act of translating an **idea** for a solution into a **clear set of instructions/expressions** in a language that can be **interpreted and executed by a computer**.

To be **effective** as a programmer you need to not only know the language well, but you also need to be able to “speak” or “translate” the ideas into the language clearly and correctly.

Problem Solving Example:

A first “program”

Introduction:

Many encoding programs (encryption, video, sound, etc.) rely on properties of division.

Task:

Write a **predicate** (function that results in “true” or “false”) that decides if an integer d divides evenly into another integer n . For example:

isDivisor(d , n) is true if ...

Solution Language (#1)

Integer operations: $+$, $-$, \times , \div , modulo (sometimes called modulus)

Relational operators: $=$, \neq , $<$, $>$, \leq , \geq

Integer Constants: 0, 1, 2, 3, . . .

Conditional Definition: if

Problem Solving: A second attempt...

Solution Language (#2)

- Integer operations: $+$, $-$, \times , \div
- Relational operators: $=$, \neq , $<$, $>$, \leq , \geq
- Integer Constants: 0 , 1 , 2 , 3 , \dots
- Conditional Definition: if

Hint: integer division
returns integer quotient
without remainder

Things to think about:

- How easy was this to solve?
- How confident are you that the solution is correct?

What is programming?

- **program:** A set of instructions to be carried out by a computer.
- **program execution:** The act of carrying out the instructions contained in a program.
- **programming language:** A systematic set of rules used to describe computations in a format that is editable by humans.
 - This course teaches programming in a language named Java.
 - There are other languages that are similar to Java (C#, C++).
 - Some are different (Perl, PHP)

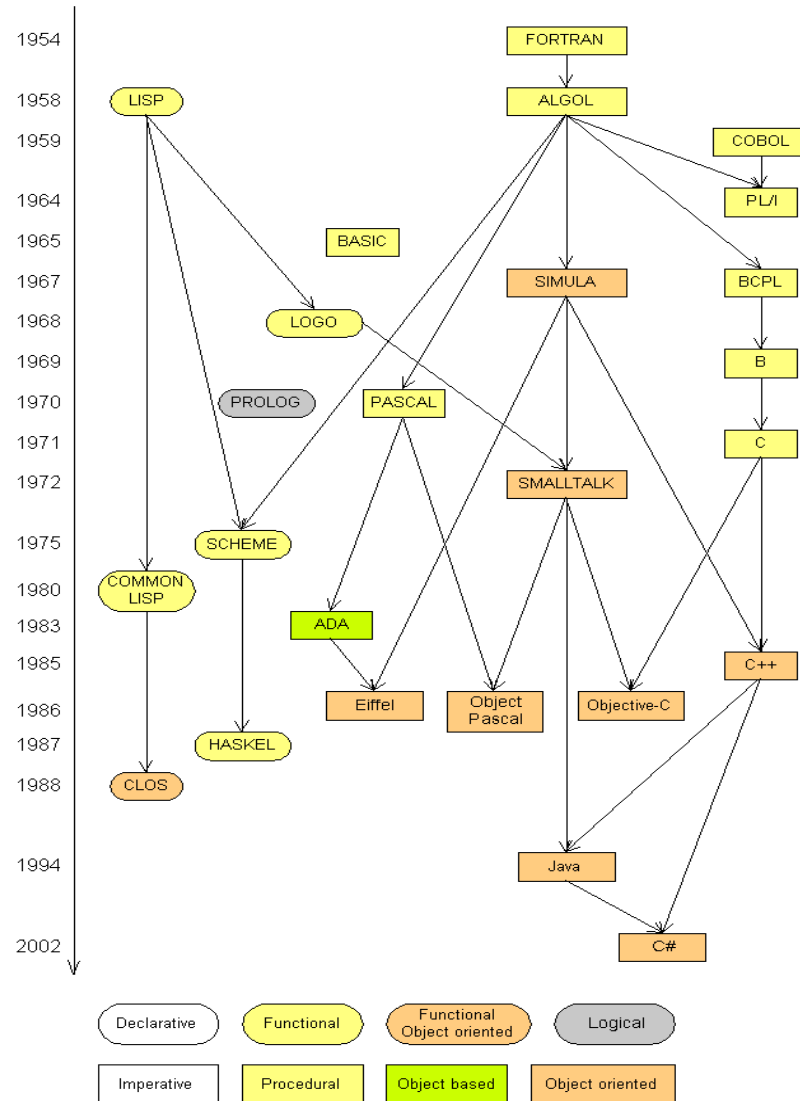


History

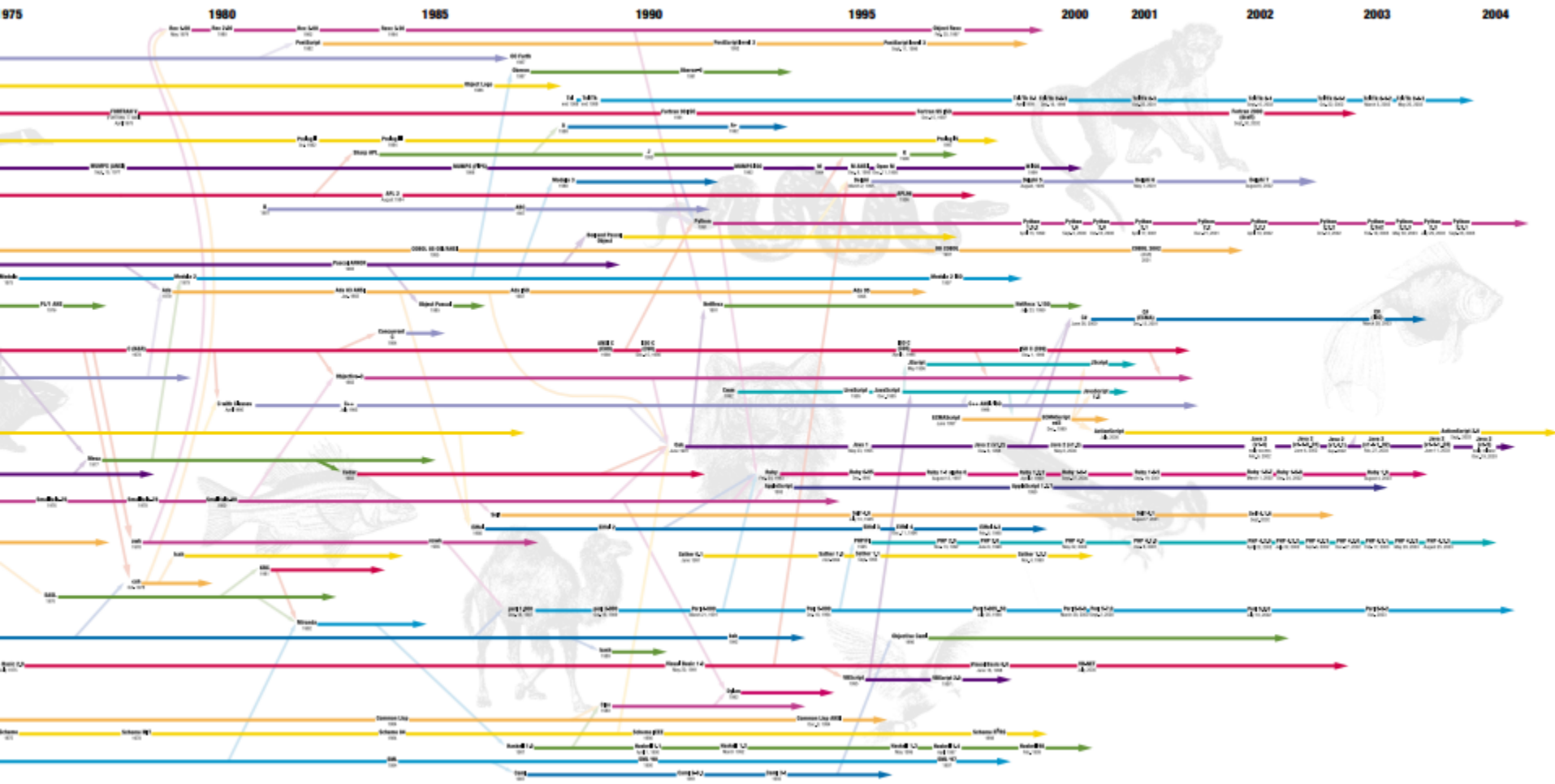
- The meaning of **computer** has changed over the past 70 years
- It used to describe a human being
- People in the picture were human computers
 - @ the NACA High Speed Flight Station, summer 1949
- Most human computers were PhD mathematicians
 - During WW2 most were women
 - Some became the first programmers (i.e., for the ENIAC (1946))
 - The first professional programmers were women.



Programming languages



Programming languages



Source: http://cdn.oreilystatic.com/news/graphics/prog_lang_poster.pdf

Basic Java programs with `println` statements

Compile/run a program

1. Write it.

- **code** or **source code**: The set of instructions in a program.

2. Compile it.

- **compile**: Translate a program from one language to another.
- **byte code**: The Java compiler converts your code into a format named *byte code* that runs on many computer types.

3. Run (execute) it.

- **output**: The messages printed to the user by a program.



A Java program

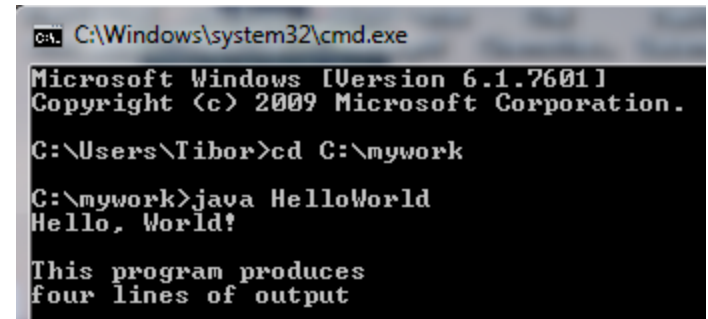
```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
        System.out.println();  
        System.out.println("This program produces");  
        System.out.println("four lines of output");  
    }  
}
```

- **Its output:**

Hello, World!

This program produces
four lines of output

- **console:** Text box into which the program's output is printed.



```
C:\Windows\system32\cmd.exe  
Microsoft Windows [Version 6.1.7601]  
Copyright (c) 2009 Microsoft Corporation.  
  
C:\Users\Tibor>cd C:\mywork  
  
C:\mywork>java HelloWorld  
Hello, World!  
  
This program produces  
four lines of output
```

Structure of a Java program

```
public class name {  
    public static void main(String[] args) {  
        statement;  
        statement;  
        ...  
        statement;  
    }  
}
```

The diagram illustrates the structure of a Java program with three callout boxes:

- class:** a program (points to **name**)
- method:** a named group of statements (points to the `main` method signature)
- statement:** a command to be executed (points to one of the **statement** lines inside the `main` method)

- Every **executable** Java program consists of a **class**,
 - that contains a **method** named `main`,
 - that contains the **statements** (commands) to be executed.

System.out.println

- A statement that prints a line of output on the console.
- Two different ways to use `System.out.println` :
 - `System.out.println("text");`
Prints the given message as output.
 - `System.out.println();`
Prints a blank line of output.

Names and identifiers

- You must give your program a name.

```
public class SpongeBob {
```

- Naming convention: capitalize each word (e.g. MyClassName)
- Your program's file must match exactly (SpongeBob.java)
 - includes capitalization (Java is "case-sensitive")

- **identifier**: A name given to an item in your program.

- must start with a letter or `_` or `$`
- subsequent characters can be any of those or a number

• **legal:** `_myName` `MrKrabs` `ANSWER_IS_42` `$patrick$`

• **illegal:** `me+u` `221B` `baker-street` `squid&ward`

Keywords

- **keyword:** An identifier that you cannot use because it already has a reserved meaning in Java.

abstract	default	if	private	this
boolean	do	implements	protected	throw
break	double	import	public	throws
byte	else	instanceof	return	transient
case	extends	int	short	try
catch	final	interface	static	void
char	finally	long	strictfp	volatile
class	float	native	super	while
const	for	new	switch	
continue	goto	package	synchronized	

Syntax

- **syntax:** The set of legal structures and commands that can be used in a particular language.
 - Every basic Java statement ends with a semicolon ;
 - The contents of a class or method occur between curly braces, that is between { and }
- **syntax error (compiler error):** A problem in the structure of a program that causes the compiler to fail, due to:
 - Missing semicolon
 - Too many or too few { } braces
 - Illegal identifier for class name
 - Class and file names do not match (Case sensitive!)
 - ...

Syntax error example

```
1 public class HelloWorld {  
2     pooblic static void main(String[] args) {  
3         System.owt.println("Hello, World!")_  
4     }  
5 }
```

- Compiler output:

```
HelloWorld.java:2: <identifier> expected  
    pooblic static void main(String[] args) {  
      ^
```

```
HelloWorld.java:3: ';' expected  
    }  
    ^
```

```
2 errors
```

- The compiler shows the line number where it found the error.
- The error messages can be hard to understand yet they will make more sense over time with experience!