Engineering Software Development

Mark A. Austin

University of Maryland

austin@umd.edu
ENCE 688R, Spring Semester 2023

February 6, 2023

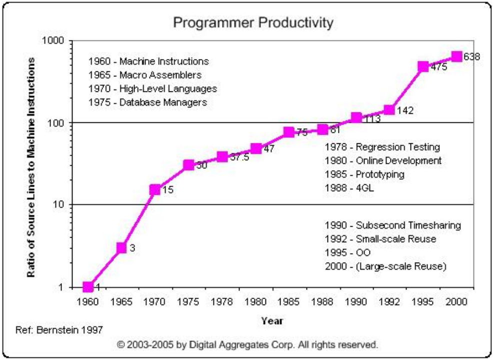
Overview

Quick Review

Pathway to Improved Programmer Productivity

Pathway Forward

Major increases in designer productivity have nearly always been accompanied by new methods for solving problems at higher levels of abstraction.



Evolution of Computer Languages

Computer Languages. Formal description – precise grammar – for how a problem can be solved.

Evolution. It takes about a decade for significant advances in computing to occur:

Capability	1970s	1980s	1990s
Users	Specialists	Individuals	Groups
Usage	Numerical computations	Desktop computing	E-mail, web, file transfer.
Interaction	Type at keyboard	Screen and mouse	audio/voice.
Languages	Fortran, C	MATLAB	HTML, Java

Popular Computer Languages

Tend to be designed for a specific set of purposes:

- FORTRAN (1950s – today). Stands for formula translation.
- C (early 1970s – today). New operating systems.
- C++ (early 1970s – today). Object-oriented version of C.
- MATLAB (mid 1980s – today). Stands for matrix laboratory.
- Python (early 1990s – today). A great scripting language.
- HTML (1990s – today). Layout of web-page content.
- Java (1994 – today). Object-Oriented language for network-based computing.
- XML (late 1990s – today). Description of data on the Web.

Problem Solving with Computers

Problem Solving with Computers

Develop Model of System Context:

- What is the context within which the software will operate?

Operations Concept:

- What is the required system functionality?
- What are the system inputs and outputs?
- What will the system do in response to external stimuli?

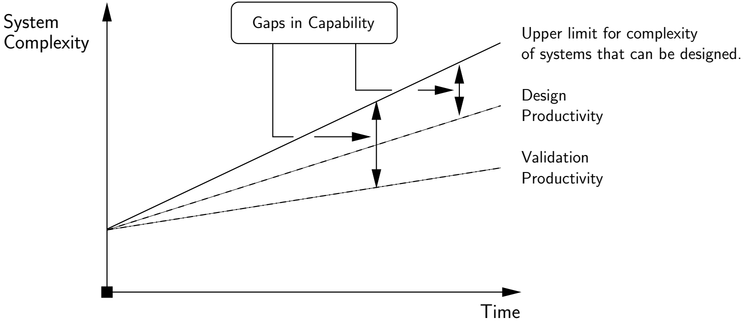
Requirements:

- What requirements are needed to ensure that the system will operate as planned?
- How will the software be written, tested, maintained?

Strategies for Handling Complexity

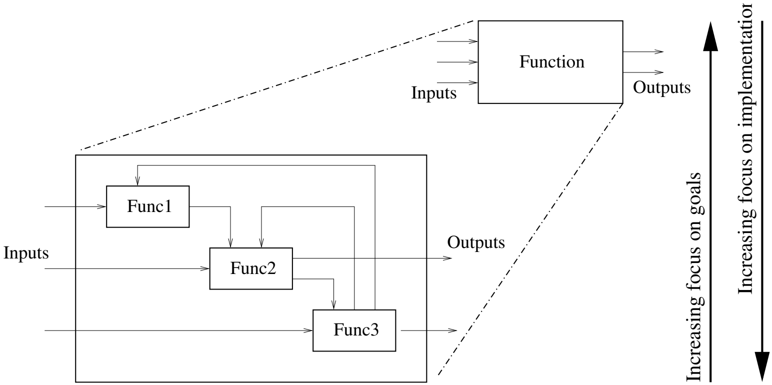
Productivity Concerns

System designers and software developers need to find ways of being more productive, just to keep the duration and economics of design development in check.



Strategies for Handling Complexity

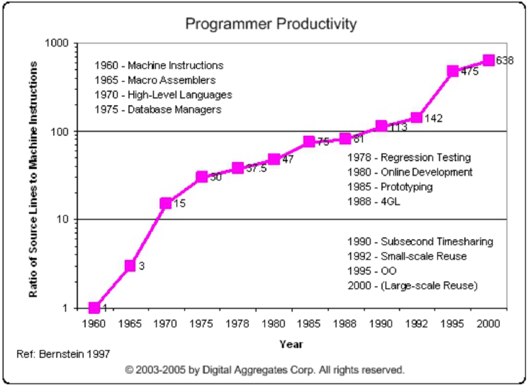
Simplify models of funtionality by decomposing high-level functions into networks of lower-level functionality:

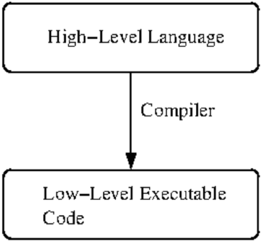


Strategies for Handling Complexity

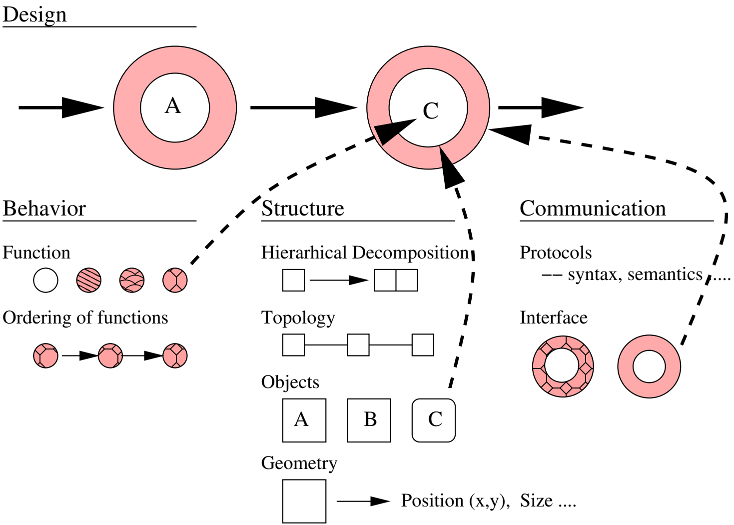
Create High-Level Description of Solution:

Increasing System Complexity: Software programmers need to find ways to solve problems at high levels of abstraction.





Separation of Concerns



Separation of Concerns

Models of System Structure:

- Specify how a system (including software) will solve a problem.
- Includes development of functional hierarchies and network structures.

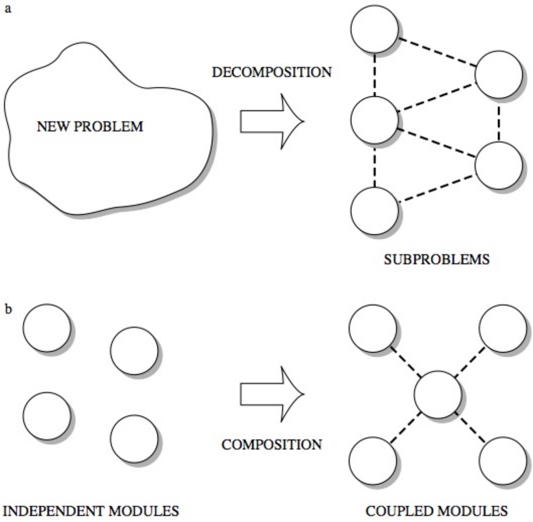
Models of System Behavior:

- Specify what the system (including software) will do.
- Includes top-level functionality, inputs and outputs, order of function execution.

Models of System Communication:

- Specification for how subsystems will communicate.
- Includes specification of interfaces and protocols for communication.

Top-Down and Bottom-Up Design



Top-Down and Bottom-Up Design

Top-Down Development:

- Can customize a design to provide what is needed and no more.
- Start from scratch implies slow time-to-market.

Bottom-up Development:

- Reuse of components enables fast time-to-market.
- Reuse of components improves quality because components will have already been tested.
- Design may contain many features that are not needed.

This Class:

- Extensive use of software libraries (e.g., collections).

Modeling

System Behavior

Abstractions for Modeling System Behavior

Program Control → System Behavior:

Behavior models coordinate a set of what we will call steps.

Two questions for each step:

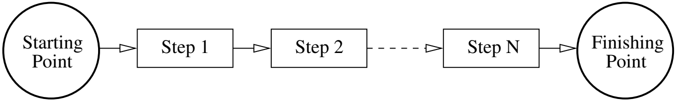
- When should each step be taken?
- When are the inputs to each step determined?

Abstractions that allow for the ordering of functions include:

- Sequence constructs,
- Branching constructs,
- Repetition/looping constructs,
- Concurrency constructs.

Abstractions for Modeling System Behavior

Sequencing of Steps in an Algorithm:
Which functions must precede or succeed others?

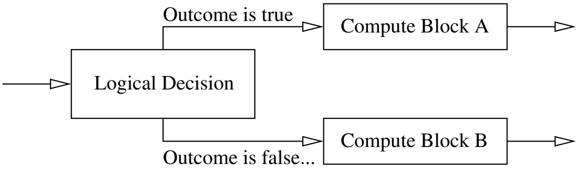


The textual/pseudocode counterpart is:

```
Starting Point
   Step 1.
   Step 2.
   Step 3.
   ......
   Step N.
Finishing Point
```

Abstractions for Modeling System Behavior

Selection Constructs: Capture choices between functions



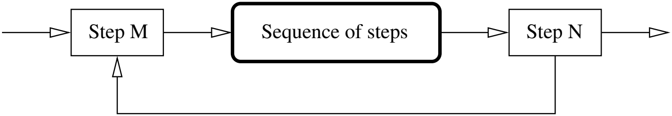
Languages need to support evaluation of relational and logical expressions.

```
Question: Is 4 greater than 3?
Expression: 4  > 3 ... evaluates to ... true.

Question: Is 4 equal to 3?
Expression: 4 == 3 ... evaluates to ... false.
```

Abstractions for Modeling System Behavior

Repetition/Looping Constructs:



Repitition constructs want to know:

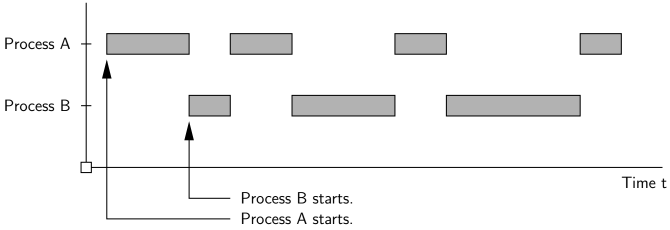
- Which functions can be repeated as a block?

Abstractions for Modeling System Behavior

Ordering of Functions: Concurrency

Most real-world scenarios involve concurrent activities. The key challenge is sequencing and coordination of activities to maximize a system's performance.

Example 1. Running multiple threads of execution on one processor:



Interpreted and Compiled Languages

Interpreted Programming Languages

Interpreted Programming Languages:

- High-level statements are read one by one, and translated and executed on the fly (i.e., as the program is running).

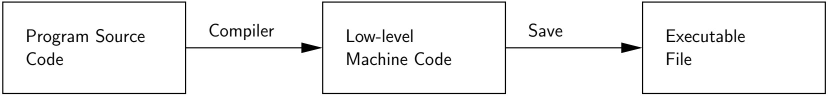
Examples:

- HTML and XML.
- Visual Basic and Javascript.

Scripting languages such as Tcl/Tk and Perl are interpreted. Python and Java are both interpreted and compiled.

Compiling the Program Source Code

A compiler translates the computer program source code into lower level (e.g., machine code) instructions.



High-level programming constructs (e.g., evaluation of logical expressions, loops, and functions) are translated into equivalent low-level constructs that a machine can work with.

Examples: C and C++.

Benefits of Compiled and Interpreted Code

Benefits of Compiled Code:

- Compiled programs generally run faster than interpreted ones.
- This is because an interpreter must analyze each statement in the program each time it is executed and then perform the desired action.

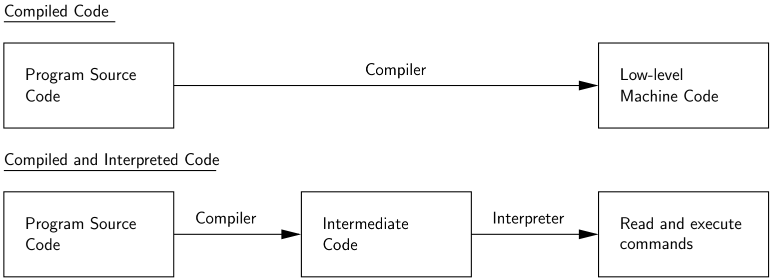
Benefits of Interpreted Code:

- Interpreted programs can modify themselves by adding or changing functions at runtime.
- Cycles of application development are usually faster than with compiled code because you don't have to recompile your application each time you want to test a small section.

Compiled and Interpreted

Modern Interpreter Systems

Transform source code into a lower-level intermediate format.
Interpreter then executes commands.



Examples: MATLAB, Java and Python.

Implementation

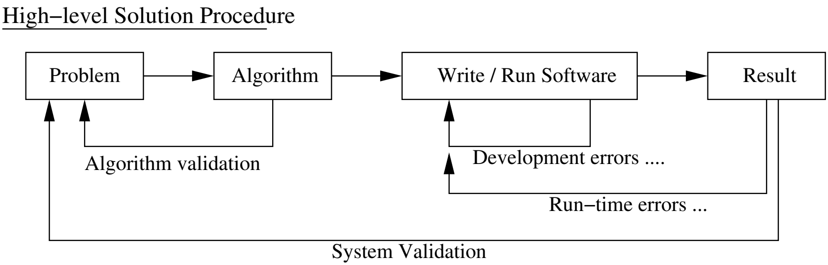
(Writing the Code)

Problem Solving with Computers

Computer Programming

Learn how to translate an algorithm into a set of instructions that a computer can understand.

High-Level Problem Solving Procedure:



Implementation

Writing the Program Source Code

When you write the source code for a computer program, all you are doing is using text to fill-in the details of programming templates.

Details of the syntax will vary from one language to another, e.g.,

```
Branching Construct in Java       Branching Construct in Python
==============================================================

if ( i < 3 ) {                    if i < 3:
 .... do something ....              .... do something ....
} else {                          else:
  .... do something else ....        .... do something else ....
}
==============================================================
```

Program Development with Python

A Little History

Origins of Python

The Python programming language was initially written by Guido van Rossum in the late 1980s and first released in the early '90s. Its design borrows features from C, C++, Smalltalk, etc.

The name Python comes from Monty Python's Flying Circus.

Version 0.9 was released in February 1991. Fast forward to 2020, and we are up to Version 3.8.

What is Python?

Features:

- Designed for quick-and-dirty scripts, reusable modules, very large systems.
- Object-oriented. Very well-designed. Well documented.
- Large library of standard modules and third-party modules.
- Works on Unix, Mac OS X and Windows.
- Python is both a compiled and interpreted language. Python source code is compiled into a bytecode format.
- Integration with external C and Java code (Jython).

What is Python?

Strengths of Python:

- Open source. Compared to C and Java, it's easy to learn.
- Provides an approximate superset of MATLAB functionality.
- Modern language with good support for object-oriented program development.

Third-Party Modules:

- NumPy is a language extension that defines the numerical array and matrix type and basic operations on them.
- SciPy uses numpy to do advanced math, signal processing, optimization, statistics, etc.
- Matplotlib provides easy-to-use plotting Matlab-style.

First Program: Evaluate and Plot Sigmoid Function

Problem Description

In neural network models, the sigmoid function:

$$\sigma(x) = \left[\frac{1}{1 + e^{-x}} \right]. \quad (1)$$

serves as an activation. Our first program evaluates and plots $\sigma(x)$ over the range x $\in [-10, 10]$.

Running the Program

From the terminal window, simply type:

```
prompt >> python3 TestSigmoidFunction.py
```

First Program: Evaluate and Plot Sigmoid Function

The Python interpreter/compiler will complain if one or more of the required packages (e.g., matplotlib) are missing.

Use pip to install missing Python Packages

The standard package-management system used to install and manage software packages written in Python is called pip (or maybe pip3).

Example: And installation is easy!

```
prompt >> pip3 install numpy
prompt >> pip3 install matplotlib
```

To get a list of installed packages:

```
prompt >> pip3 list
```

Program Source Code

```
# ========================================================
# TestSigmoidFunction.py: Evaluate/plot sigmoid function.
#
# Written by: Mark Austin                 September, 2020
# ========================================================

import math
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

# define sigmoid function ...

def sigmoid (x):
    return 1/(1 + math.exp(-x))

# main method ...

def main():
    print("--- Enter TestSigmoidFunction.main() ...");
    print("--- ================================= ...");

    # Part 1: Evaluate and print sigmoid function

    xvalues = list( np.arange( -10.0, 10.0, 0.5 ) );
    for x in xvalues:
       print ("--- sigmoid({:6.2f}) --> {:14.10f}".format(x, sigmoid(x)));

    # Part 2: Create list of sigmoid(x) values ...
```

Program Source Code

```
    # Part 2: Create list of sigmoid(x) values ...

    yvalues = []
    for x in xvalues:
       yvalues.append( sigmoid(x) );

    # Part 3: Organize and display plot ...

    fig, ax = plt.subplots()
    ax.plot( xvalues, yvalues )
    ax.set(xlabel='x', ylabel='sigmoid(x)',
           title='Plot sigmoid(x) vs x')
    ax.grid()

    # display and save plot ...

    plt.show()

    fig.savefig("sigmoid-plot.jpg")

    print("--- ================================ ...");
    print("--- Leave TestSigmoidFunction.main() ...");

# call the main method ...

main()
```

Textual Output

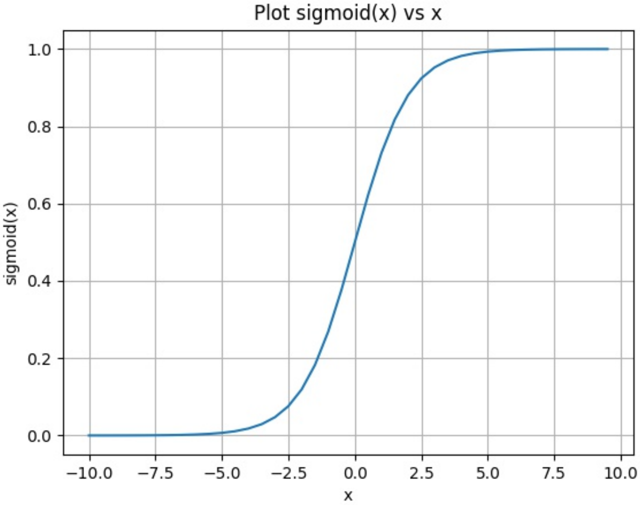
The abbreviated textual output is:

```
--- Enter TestSigmoidFunction.main()      ...
--- ======================================== ...
--- sigmoid(-10.00) -->   0.0000453979
--- sigmoid( -9.50) -->   0.0000748462
--- sigmoid( -9.00) -->   0.0001233946
--- sigmoid( -8.50) -->   0.0002034270
--- sigmoid( -8.00) -->   0.0003353501

... lines of output removed ...

--- sigmoid(  7.50) -->   0.9994472214
--- sigmoid(  8.00) -->   0.9996646499
--- sigmoid(  8.50) -->   0.9997965730
--- sigmoid(  9.00) -->   0.9998766054
--- sigmoid(  9.50) -->   0.9999251538
--- ======================================== ...
--- Leave TestSigmoidFunction.main()      ...
```

Graphical Output

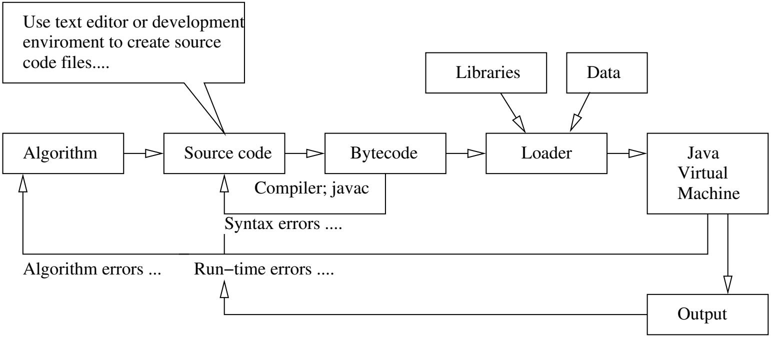


Program Development with Java

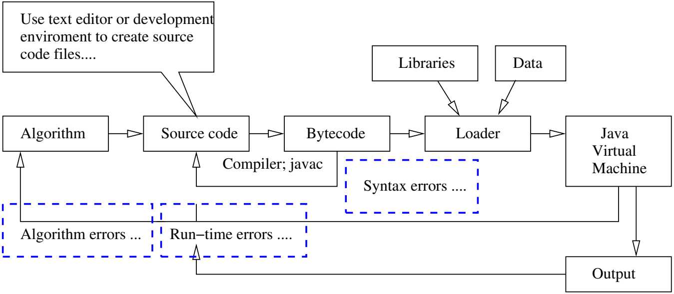
Flowchart for Software Development in Java

Step-by-Step Procedure:

1. Write, compile, fix, run, fix, run, validate → success!



First steps: Fixing mistakes!



1. Syntax Errors: Check your typing ...
2. Runtime Errors: Program runs, but you have divide by zero and/or NaNs, etc.
3. Algorithm Errors: Does your program solve the right problem?

Program Development with Java

Strengths of Java:

- Java is both a compiled and interpreted language. Java source code is compiled into a bytecode format.
- Bytecodes are the lowest possible instruction format that remain architecture neutral. As a result, the bytecode can travel across the Internet and execute on any computer that has a Java Virtual Machine.
- Java is an object-oriented language. Implementation details are made efficient by exploiting the relationship among objects.
- Language provides very good support for building large systems that will work.

Program Development with Java

Weaknesses of Java:

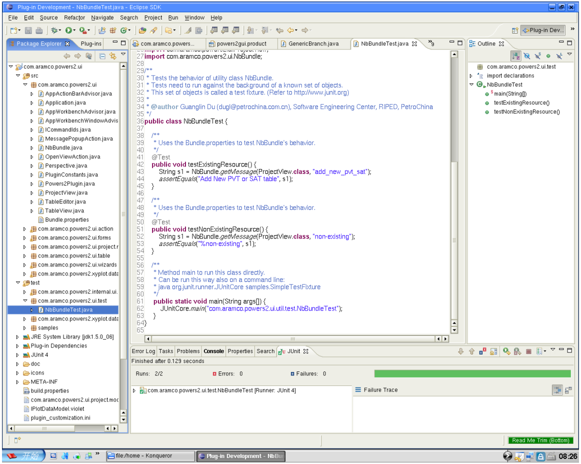
- For the solution of small problems, more lines of code than with Python, Matlab.
- There's a lot to learn, especially if you want to become really skilled at developing software in Java.

Remark:

- If you want to become really skilled at developing software in Python, there's also a lot to learn.
- Sorry, there is no free lunch.

Integrated Development Environments for Java

Eclipse is an integrated software development tool (or IDE) for Java Software Development.



First Program: Evaluate and Plot Sigmoid Function

Problem Description

In neural network models, the sigmoid function:

$$\sigma(x) = \left[\frac{1}{1 + e^{-x}} \right]. \quad (2)$$

serves as an activation. Our first program evaluates and plots $\sigma(x)$ over the range $x \in [-10, 10]$.

Compiling the Program

From the terminal window, simply type:

```
prompt >> javac TestSigmoidFunction.java
```

First Program: Evaluate and Plot Sigmoid Function

The Java compiler will complain if one or more of the required packages (e.g., math library) are missing and/or syntax errors are detected.

The files before and after compilation are as follows:

```
Before Compilation               After Compilation
--------------------------------------------------------
TestSigmoidFunction.java         TestSigmoidFunction.java
                                 TestSigmoidFunction.class
```

Running the Program

```
prompt >> java TestSigmoidFunction
```

Program Source Code

```
/*
 * ===========================================================
 * TestSigmoidFunction.java: Evaluate sigmoid function.
 *
 * Written by: Mark Austin                      September, 2020
 * ============================================================
 */

import java.lang.Math.*;

public class TestSigmoidFunction {

   // Main method ...

   public static void main (String args[] ) {
      System.out.printf("--- Enter TestSigmoidFunction.main()   ...\n");
      System.out.printf("--- ================================== ...\n");

      // main loop to evaluate and print sigmoid function ...

      for ( double x = -10.0; x <= 10.0; x = x + 0.5 ) {
          System.out.printf("--- sigmoid(%6.2f) --> %14.10f \n", x, sigmoid(x) );
      }

      System.out.printf("--- ================================== ...\n");
      System.out.printf("--- Leave TestSigmoidFunction.main()   ...\n");
   }
```

Program Source Code

```
    // Method to compute sigmoid function ...

    public static double sigmoid ( double x ) {
       return 1/(1 + Math.exp(-x));
    }
}
```

Points to note:

- The class TestSigmoidFunction must be located in a file called TestSigmoidFunction.java. During the compilation process, Java uses this one-to-one association to find classes.
- The program has two user-defined methods: main() and sigmoid().
- The statement import java.lang.Math.* makes all of the constants and methods in the math library available to this program. The program calls the exponential function.

A few more points:

- The keyword public is used in three places. It specifies that anyone can call the methods main() and sigmoid(), and that the class TestSigmoidFunction will also be public.
- The keyword static specifies that method can be called without first creating an object.
- The keyword void indicates that the method main() will not return anything.
- Variables must be declared before they can be used. Here, the variable x is a double precision floating point number.
- The method sigmoid() returns a double precision floating point number of type double.
- Java uses the dot (.) to indicate inside. The phrase System.out.printf() calls the printf() method, which is inside the out class, which is part of the System package.

Textual Output

The abbreviated textual output is:

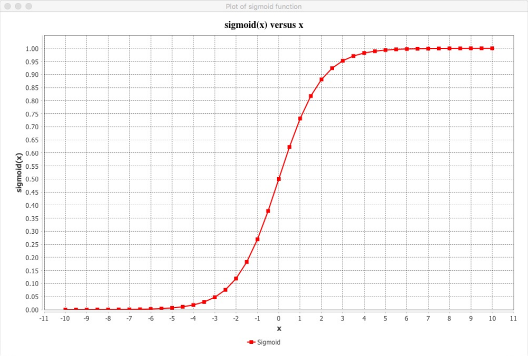
```
--- Enter TestSigmoidFunction.main()   ...
--- ================================== ...
--- sigmoid(-10.00) -->   0.0000453979
--- sigmoid( -9.50) -->   0.0000748462
--- sigmoid( -9.00) -->   0.0001233946
--- sigmoid( -8.50) -->   0.0002034270
--- sigmoid( -8.00) -->   0.0003353501

... lines of output deleted ...

--- sigmoid(  7.50) -->   0.9994472214
--- sigmoid(  8.00) -->   0.9996646499
--- sigmoid(  8.50) -->   0.9997965730
--- sigmoid(  9.00) -->   0.9998766054
--- sigmoid(  9.50) -->   0.9999251538
--- sigmoid( 10.00) -->   0.9999546021
--- ================================== ...
--- Leave TestSigmoidFunction.main()   ...
```

Creating Charts in Java

Two approaches: JFreeChart and JavaFX.



Source Code: See java-code-charts/src/ence688p/neural/