Information Communication Technologies

Lecture 9. Software Engineering

Kassymova Aizhan Bakhytzhanovna
PhD, Associate professor
a.kassymova@satbayev.university

Agenda

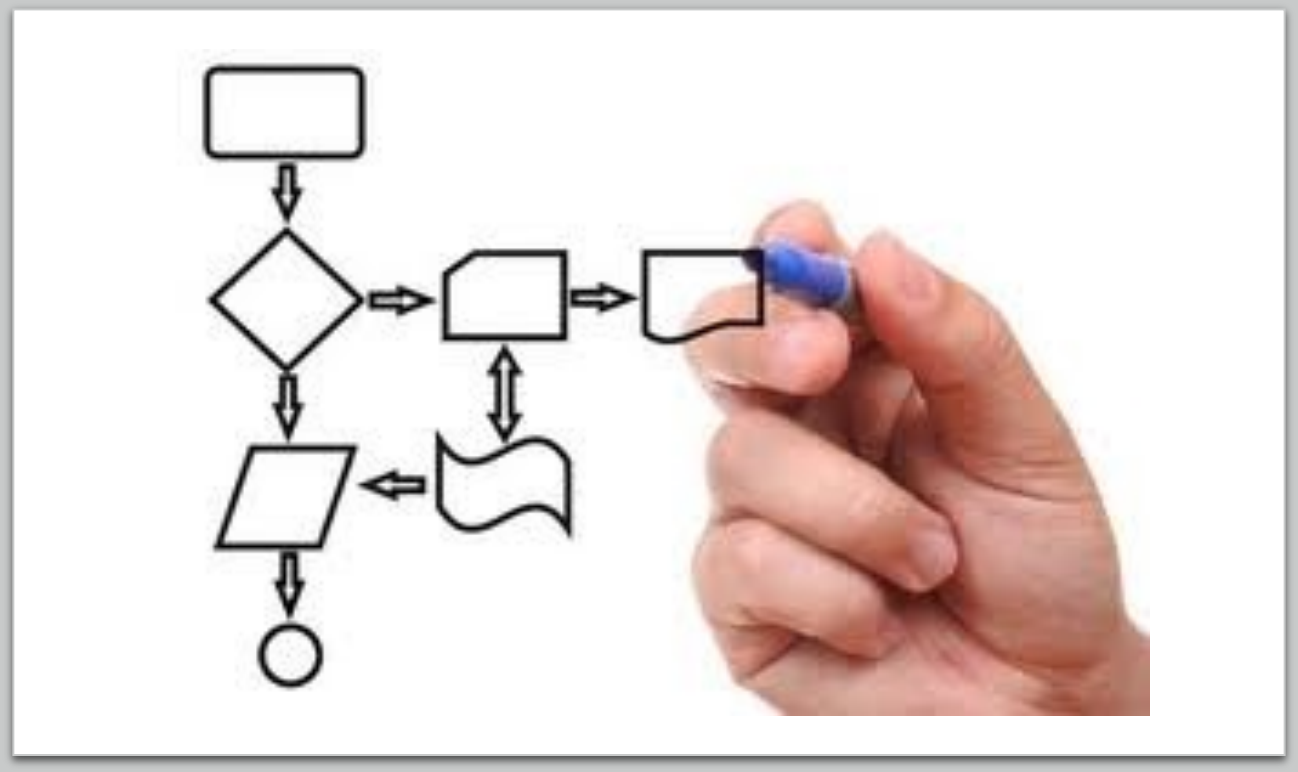
Software Engineering

Issues in Large-Scale Software Development

Open Source Model

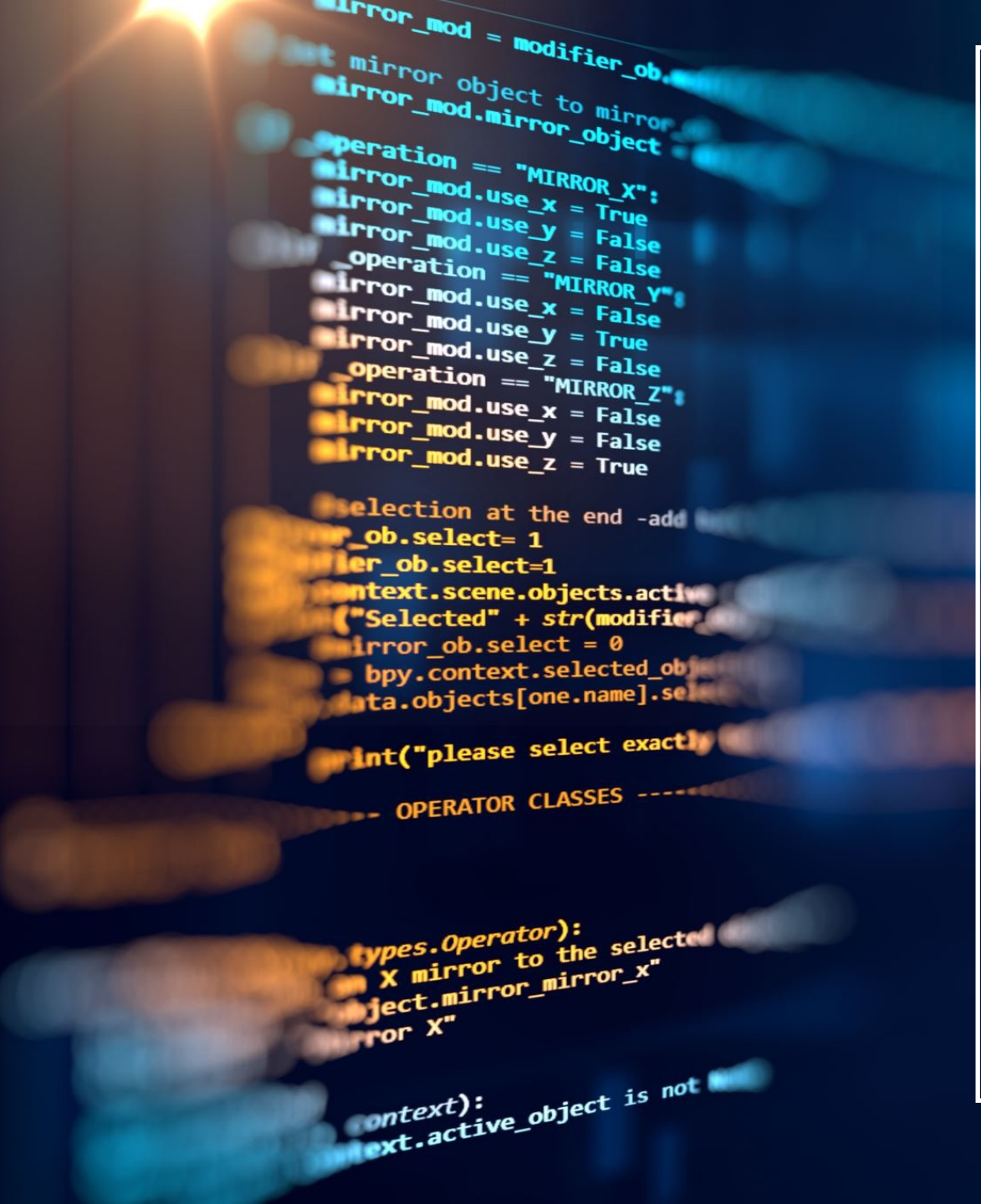
Tools for Software Creation and Management

Review Materials



Software Engineering

- Body of techniques to create, manage large scale , complex software systems by a team of programmers.



Computer Programs

- One programmer completes ***only 20 lines of code per day***.
- Most commercial programs are written by teams of programmers and take months or years to complete.
  - a large program has over 1 million lines of code

Example

Год	Версия	Строк кода
1994	Windows NT 3.5	4 000 000
1996	Windows NT 4	16 500 000
2000	Windows 2000	20 000 000
2002	Windows XP	40 000 000

Example

```
for (i=0; i<100; ++i) printf("привет"); // Сколько здесь строк кода?
```

```
for (i=0; i<100; ++i)
{
    printf("привет");
}
// Сколько здесь строк кода?
```

Example

C

```c
#include <stdio.h>

int main(void) {
   printf("Hello World");
   return 0;
}
```

Строк кода: 5

COBOL

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. HELLOWORLD.
000300
000400*
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000700 SOURCE-COMPUTER. RM-COBOL.
000800 OBJECT-COMPUTER. RM-COBOL.
000900
001000 DATA DIVISION.
001100 FILE SECTION.
001200
100000 PROCEDURE DIVISION.
100100
100200 MAIN-LOGIC SECTION.
100300 BEGIN.
100400     DISPLAY " " LINE 1 POSITION 1 ERASE EOS.
100500     DISPLAY "Hello world!" LINE 15 POSITION 10.
100600     STOP RUN.
100700 MAIN-LOGIC-EXIT.
100800     EXIT.
```

Строк кода: 17

The Problem Statement

- A ***problem statement*** defines certain elements that must be manipulated to achieve a result or goal.
- A good problem statement for a computer program:
  - ***specifies any assumptions that define scope of problem***
  - ***specifies the known information***
  - ***specifies when problem has been solved***

The Problem Statement

- An ***assumption*** is something that you accept as true so as to proceed with the program design.
- **For example**, *if you are comparing the price and quality of two pizzas, you can make the assumption that they are the same size and have the same toppings*.

The Problem Statement

- The **known information** is the information that you supply to the computer to help it solve a problem (**“givens”)**

- After identifying known information you specify how to determine when the problem has been solved.
  - **specify the outcome you expect**

Algorithms

- An ***algorithm*** is a set of steps for carrying out a task, which can be written down or implemented.
    - like a recipe

- To design an algorithm, you could record the steps that you take to solve the problem.
- Your algorithm should specify how to manipulate the information.

Expressing an Algorithm

- You can express an algorithm in:
  - **structured English**
  - **pseudocode**
  - **flowcharts**
  - **object definitions**

Structured English

- Structured English is a subset of the English language with limited selection of sentence structures.

```
IF customer has a Bank Account THEN
     IF Customer has no dues from previous account THEN
         Allow loan facility
     ELSE
       IF Management Approval is obtained THEN
         Allow loan facility
       ELSE
         Reject
       ENDIF
     ENDIF
   ELSE
     Reject
   ENDIF
```

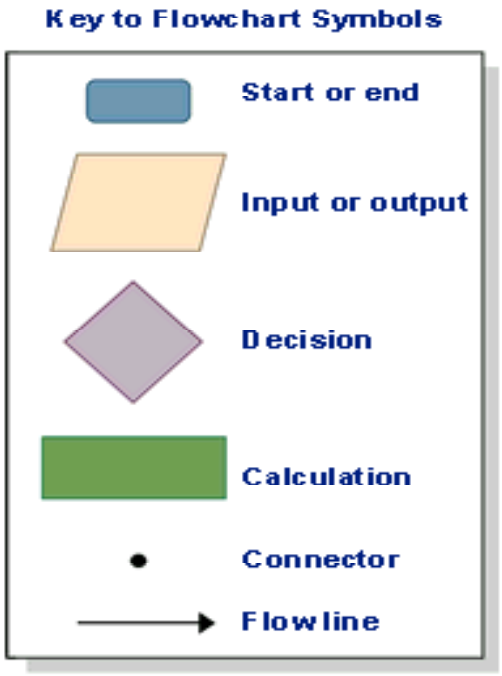
Expressing an Algorithm

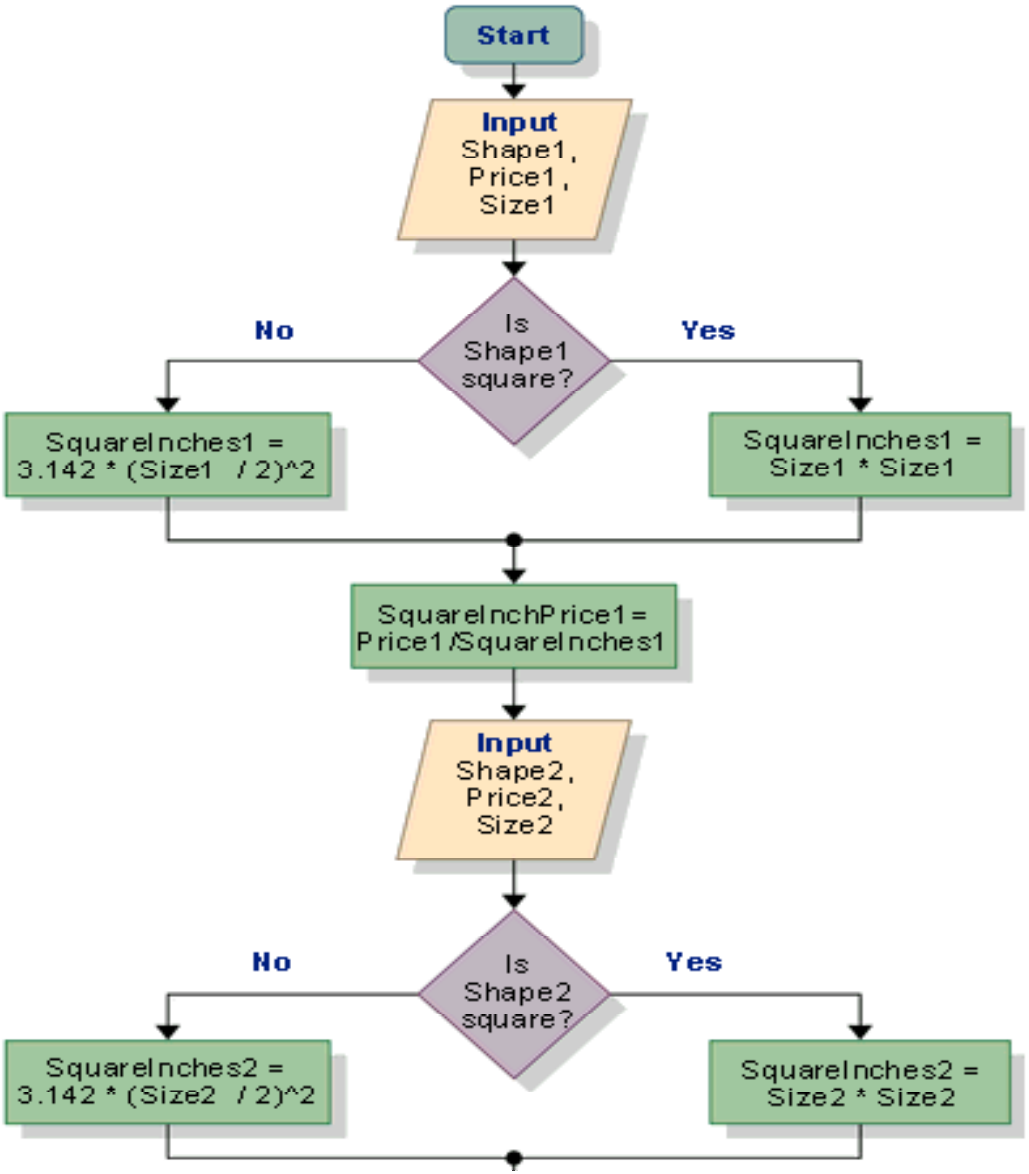
- ***Pseudocode*** is a notational system for algorithms that has been described as a “mixture of English and your favorite programming language.”
- A ***flowchart*** is a graphical representation of the way that a computer should progress from one instruction to the next when it performs a task.

```
IF <condition>
    DO action;
ELSE
    DO next action;

WHILE <condition>
    DO action;

for <variable> from <first value> to <last value> by <step>
    DO action with variable;
```

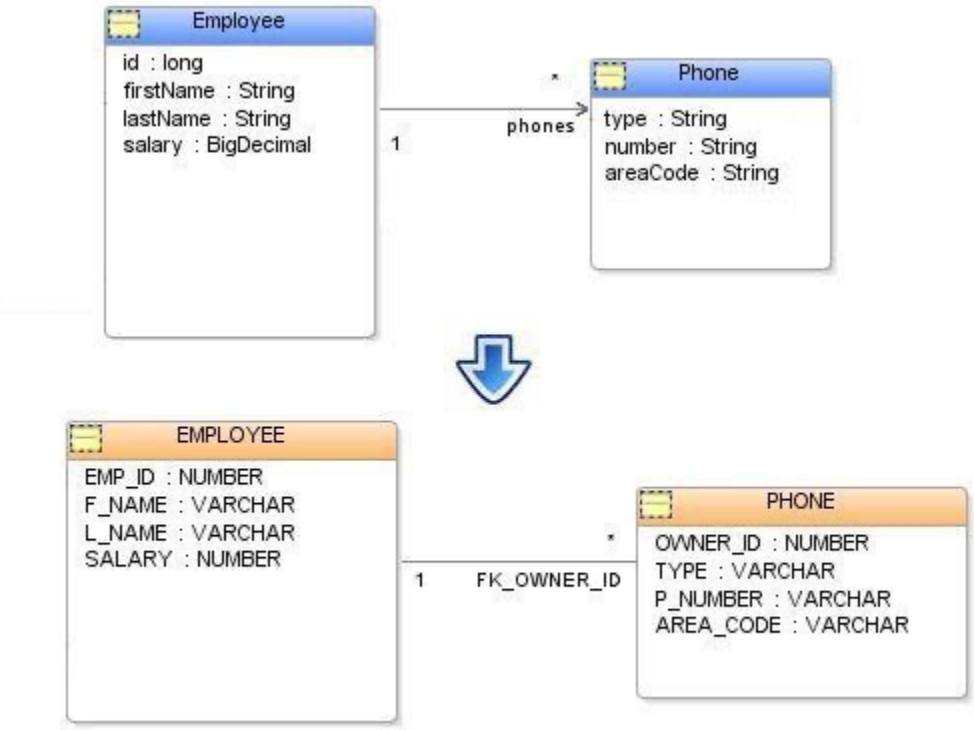
Expressing an Algorithm

An example of a flowchart

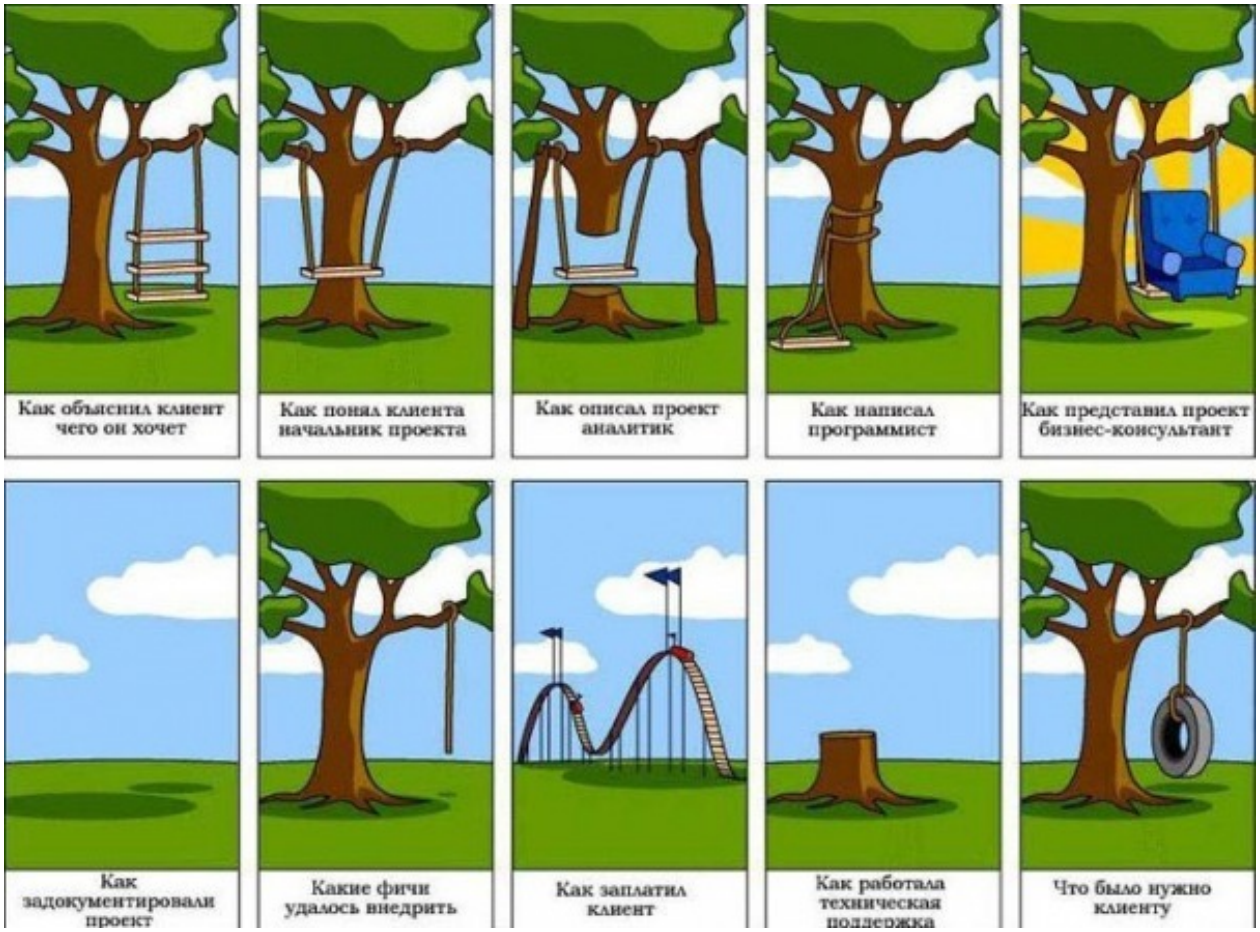




Object Definitions



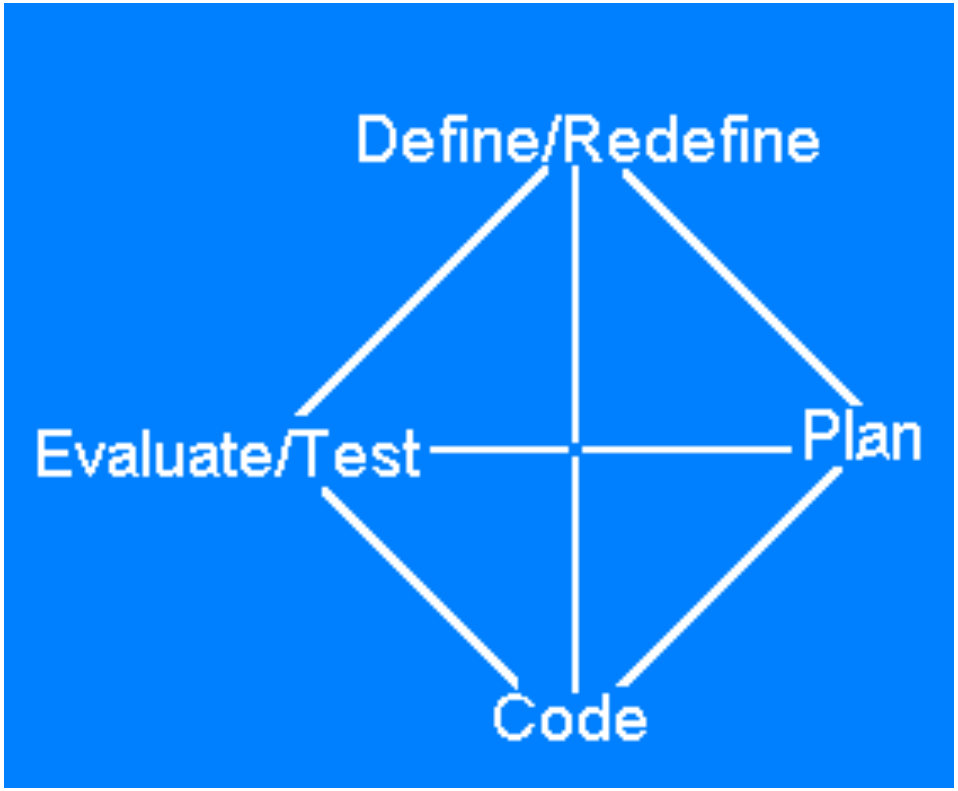
Software Development Process



Software Development Process

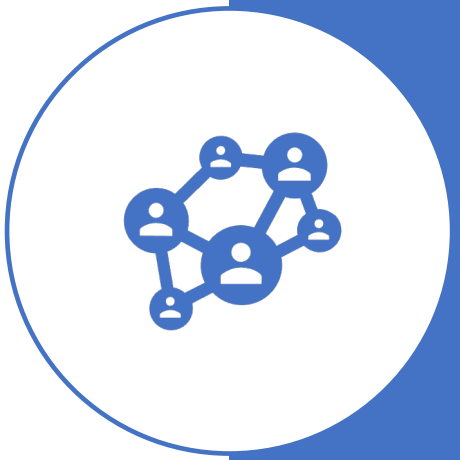
- Coding is a small part of the software process.
- Includes the recognition of the needs and the other development phases till delivery and deployment.
- They are:
  - ***People who define the needs for the software process.***
  - ***People who decide the system requirements for the software process.***
  - ***People who design and code the solution***.

Software Development Process (continued)



Define/Redefine the Solution

- Two steps involved are:
- ***Recognition of the needs :***
  - may arise from marketing or management, technical groups and it might come as a contract.
- ***Specification of the requirements***:
  - process of polling the needs of the software system.
  - Advisable to make tests and iterations with focus groups comprising of actual users.



Plan Solution

- ***Enumeration of alternatives****:*
  - Enumerate the solutions for the software system
  - It may more than one each having its own compatibility with the system, ease of operation and system costs.
- ***System Requirements:***
  - Designing the system requirements for the needs
  - Performing critical tests for the systems

Code the Solution

- ***Implementation of the solution****:*
  - Coding for the solution is written in programming languages
  - Details of method to write the code depends on the programmer

Code the Solution (continued)

- ***Programmer testing:***
  - Programmers test the solution for its correct performance
  - Group of programmers read and comment on others code in **code review session**
  - **System testing is lead by project leaders**.

Code the Solution (continued)

- ***System Acceptance:***
  - Group of individuals test the software for a period of time under simulated and real settings.

Acceptance Tests - Functionality

Evaluate and Test the Solution

- ***Test in context**:*
  - The working systems will exhibit bugs that slipped past filters.
  - Design decisions will reveal their flaws.
  - Demand for additional features or redesign for the solution.

Evaluate and Test the Solution (continued)

- ***Redesign the solution:***
  - Redesigning starts from the beginning of the solution.

Program Documentation

- ***Written documentation*** is external to the program and contains information about the program that is useful to programmers and the people who use it.

Figure 15-15 Different types of documentation have different characteristics.

	Program Manual	Reference Manual
Audience	Programmers	Users
Major focus	Problem statement Algorithm	How to install the program How to start the program How to use each feature
Writing style	Technical, using structured English, pseudocode, flowcharts	Nontechnical prose
Additional material	Printout of program code	Troubleshooting tips Diagrams of screens and menus

Program Documentation (continued)

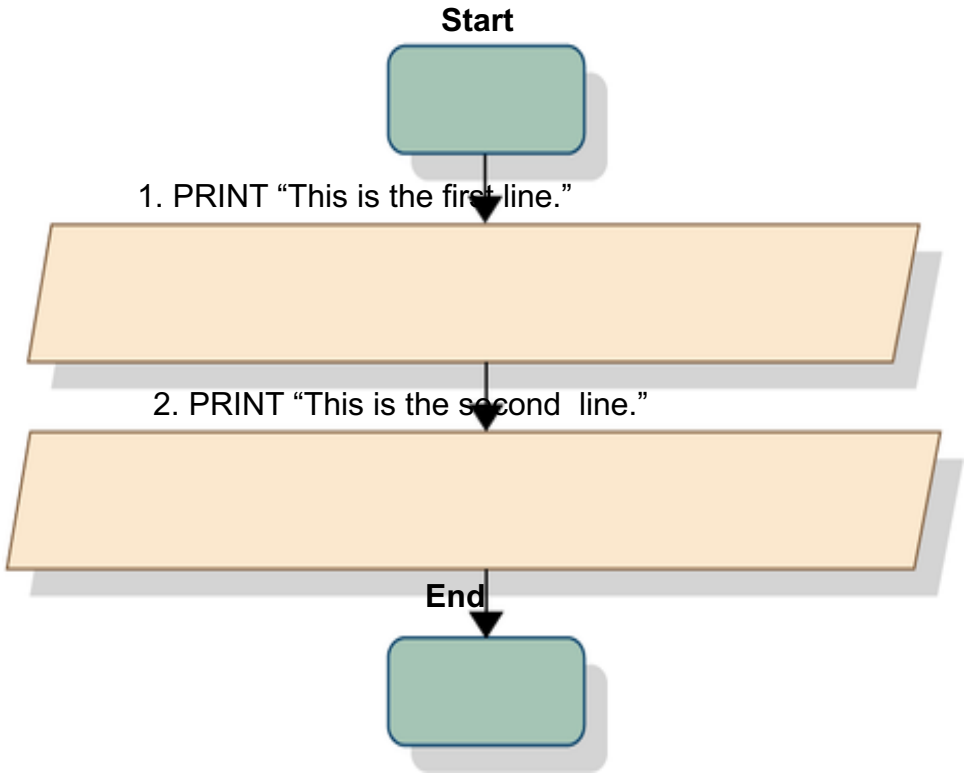
- A ***program manual*** contains any information about the program that might be useful to programmers, including the problem statement and algorithm.
  - used by programmers
- **Reference manuals** are used by people who use the program.
- A ***technical writer*** specializes in explaining technical concepts and procedures, often by simplifying complex concepts for a non-technical audience.

Coding Computer Programs

- A **problem statement** and an **algorithm** are often combined into a document called the ***program specification***.
- ***Coding*** is the process of using a computer language to express an algorithm.
  - entering commands
- A person who codes or writes computer programs is called a ***computer programmer***.

Program Sequence

- ***Sequential execution***, in which the computer performs each instruction in the order in which they appear, is the normal pattern of program execution.



Program Sequence (continued)

- ***Control structures*** are instructions that specify the sequence in which a program is executed.
- There are three types of control structures:
  - sequence controls
  - selection controls
  - repetition controls

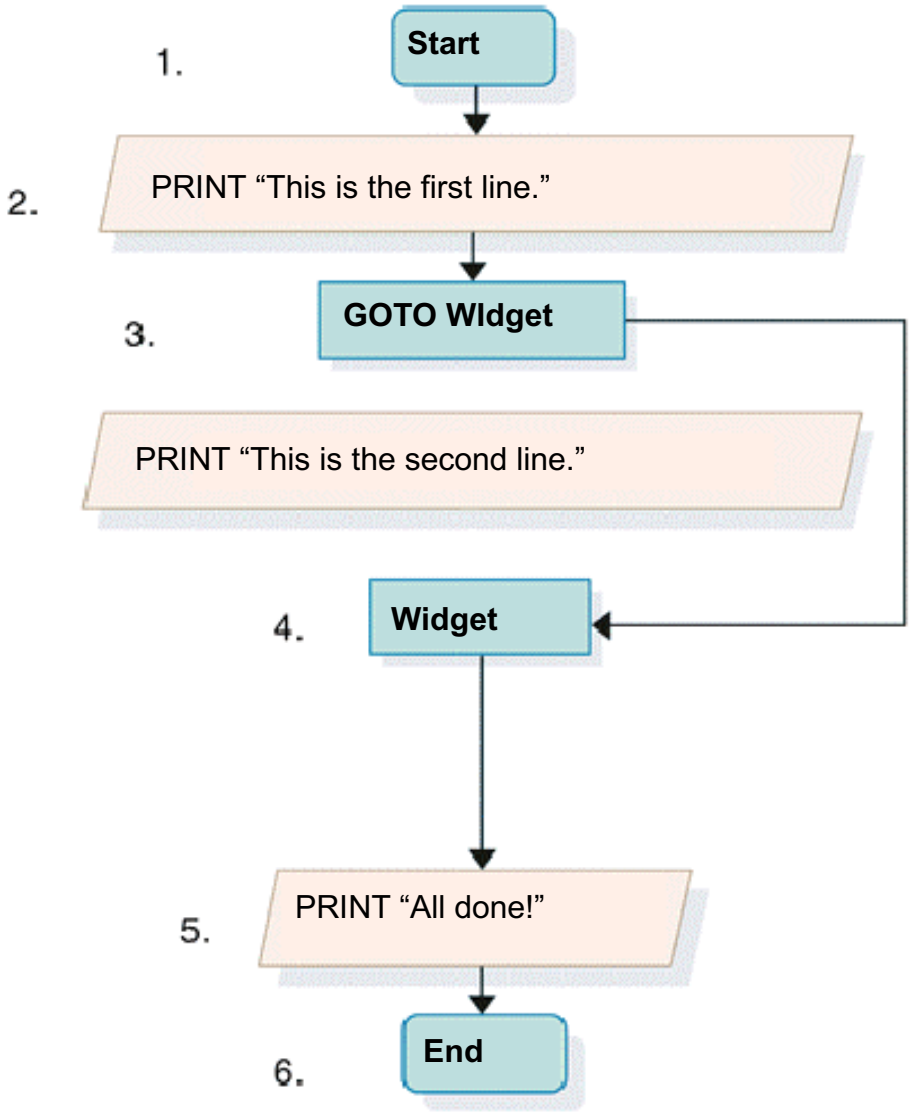
Sequence Control

- A ***sequence control structure*** changes the sequence, or order, in which instructions are executed by directing the computer to execute an instruction elsewhere in the program.

- GOTO

Sequence Control (continued)

An example of a sequence control structure



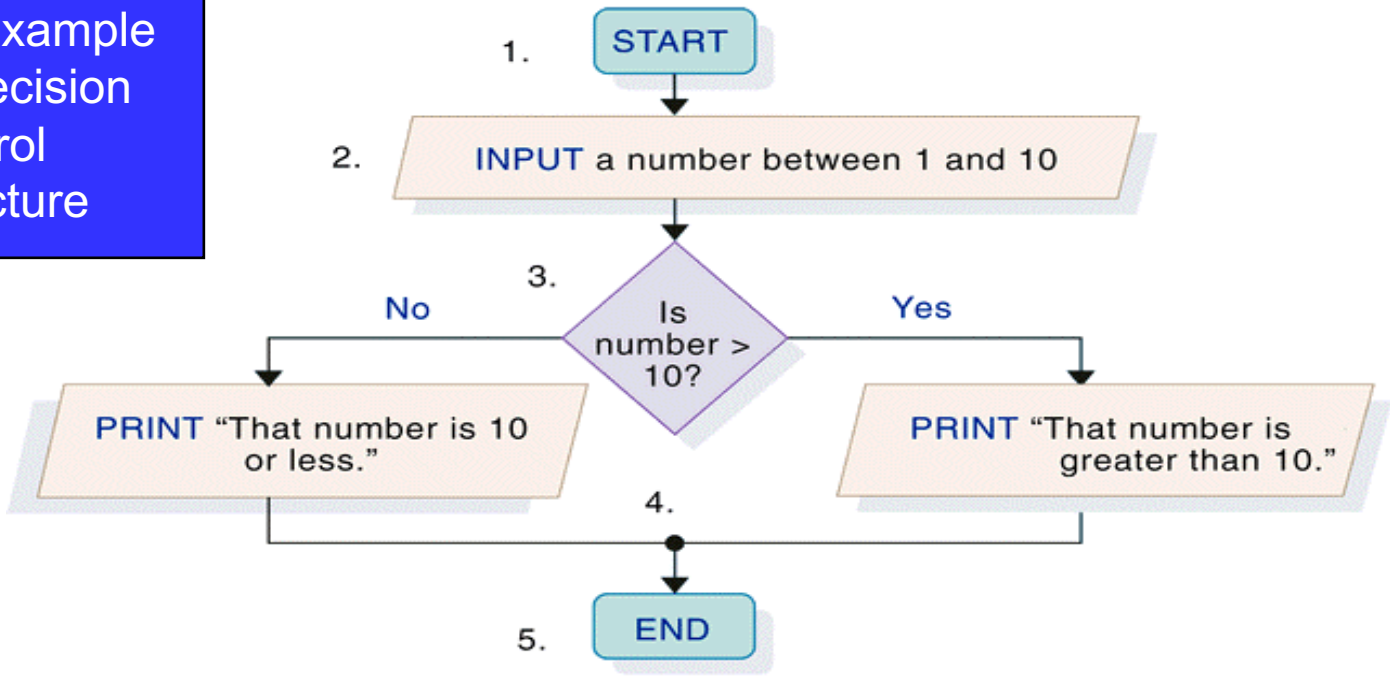
Sequence Control (continued)

- Experienced programmers prefer to use sequence controls other than GOTO.
- A ***subroutine procedure, module or function*** is a section of code that is part of a program, but not included in the main sequential execution path.

Selection Controls

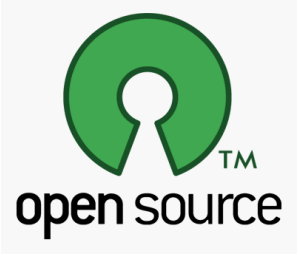
- A ***selection control structure*** (also known as a "decision structure" or a "branch") tells a computer what to do based on whether a condition is true or false.
  - an example is an IF...THEN...ELSE command

An example of decision control structure

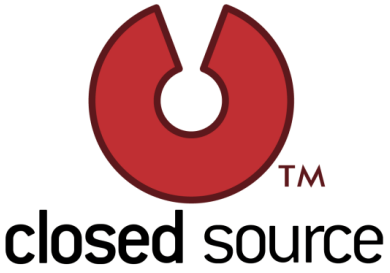


Software Models

- ***Open source*** code is available to change, copy and distribute freely.
  - Linux
  - Netscape



- ***Closed source*** code is not available to alter and protected by copyright laws
  - Certain versions of Microsoft Office applications and O/S



Open Source: India

- **India**: Bharat Operating System Solutions (BOSS) – Linux OS
- Update of Microsoft Software – 30 mln rupee

Open Source: Germany

- Government agencies have used Open Source Software: **Linux, OpenOffice, Thunderbird**
- **Problem:** Microsoft can implement support better and cheaper than the small companies that support Linux

- Avaricious pays twice (скупой платит дважды)

Open Source: Russia

- Project “Penguin”: introduction of Linux and open source software at schools and universities in Russia
- December 2010: Putin ordered to transfer “power” (government) to the Linux

Open Source: Linux/GNU

- Slackware, Debian GNU/Linux, Red Hat, Fedora, Mandriva, SuSE, Gentoo, Ubuntu.
- 60 % servers worldwide

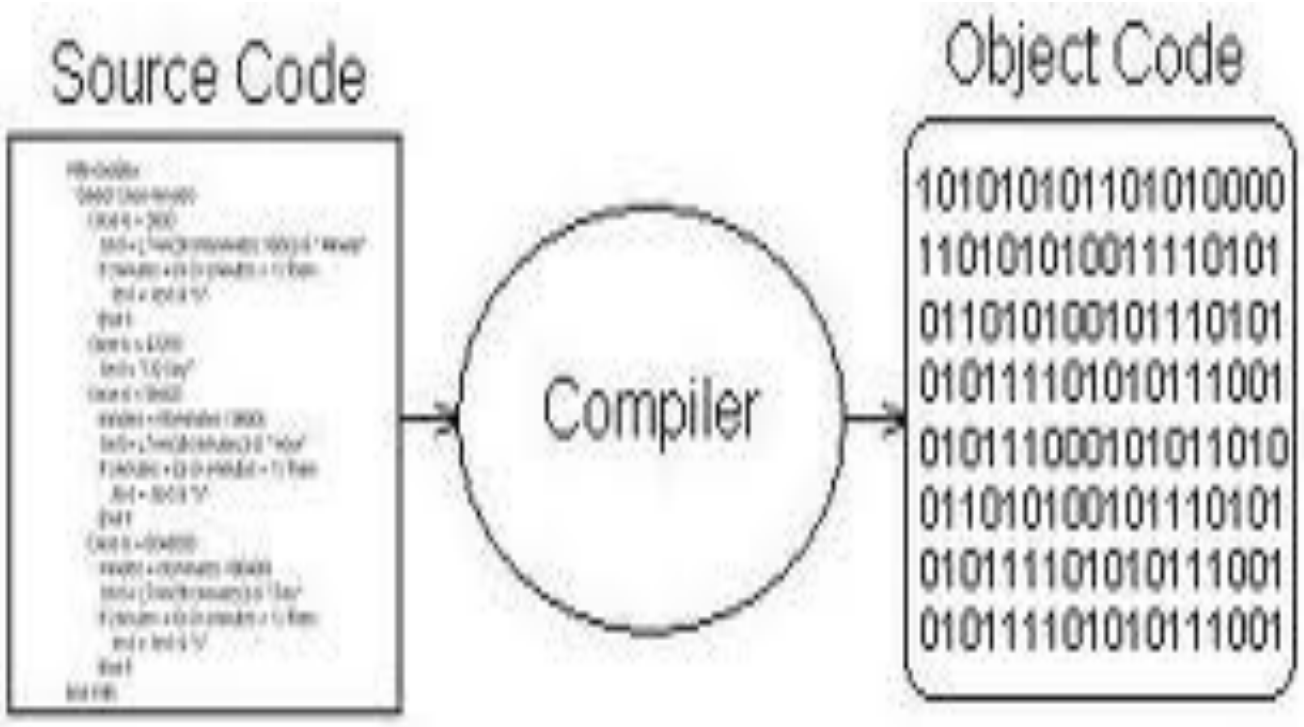
Tools for Software Creation

- A ***compiler*** translates a program written in a high-level language into low-level instructions before the program is executed.

- The commands written in a high-level language are referred to as the ***source code***.

- The resulting low-level instructions are referred to as the ***object code***.

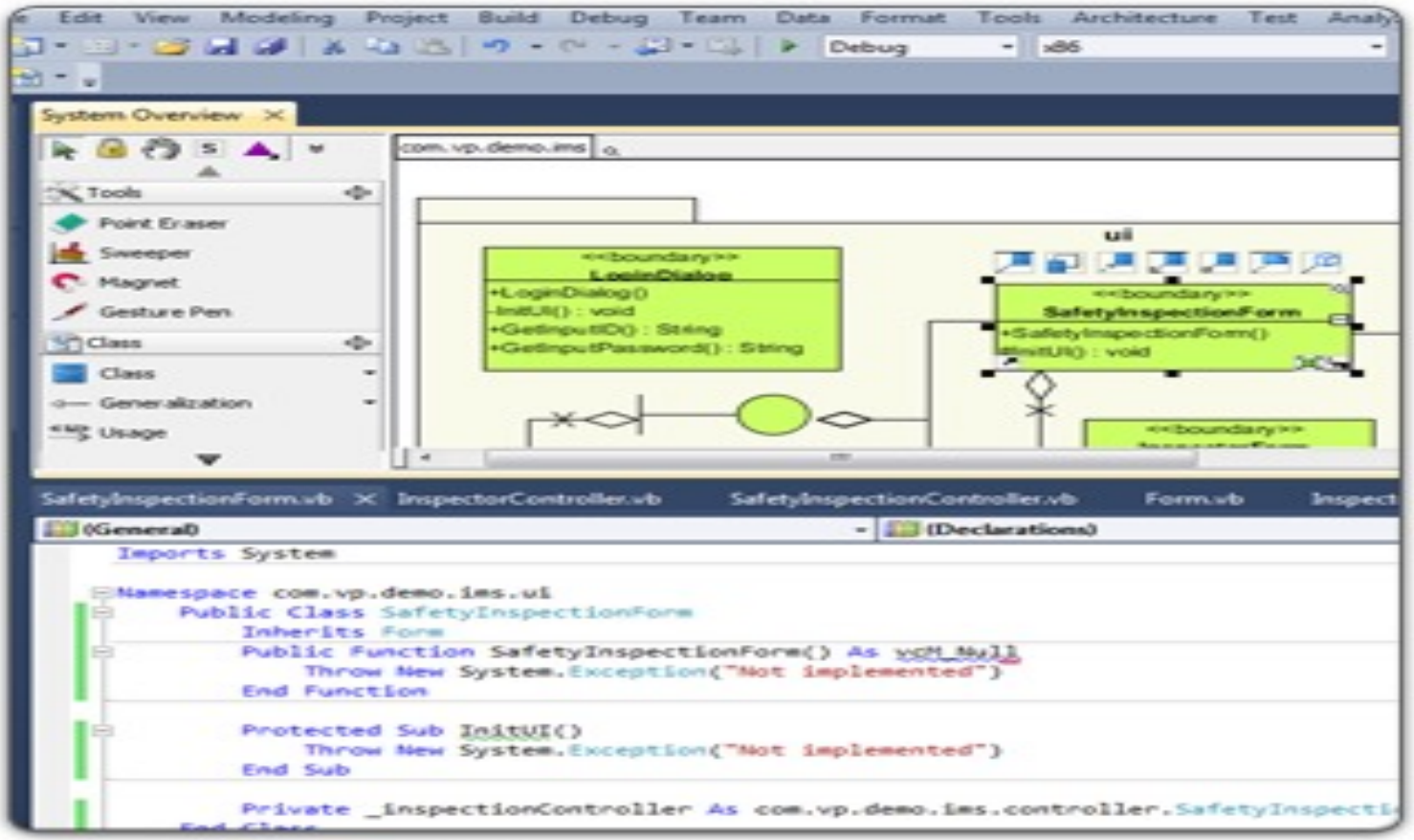
Tools for Software Creation



Tools for Software Creation (continued)

- ***Editors*** are text based applications that allow creation of source code
- ***Debugger*** is a tool used to find errors in source code during executing or running the program
- ***Integrated Development Environment*** (IDE) is a tool that aids in creation of source code – JPad is an IDE for JAVA

Integrated Development Environment



Microsoft® Visual Studio®

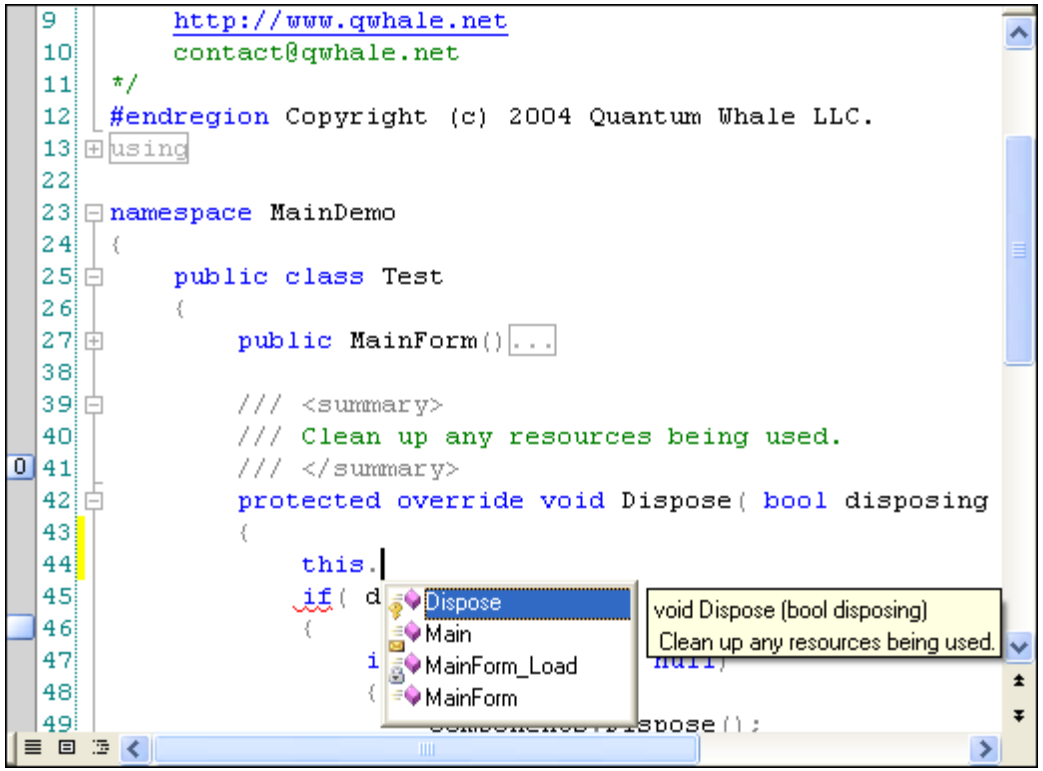
Tools for Managing Software Creation and Management

- **Programming Tools**: Special applications used to write software.
  - Editors
  - Compilers
  - Debuggers
  - Integrated Development Environments (IDEs)

Editors

- Other editors supports specific programming language
  - They have automation built in them and they **track the syntax error** in the program
  - they also track the **semantic errors** like checking the data type of the variables.

Editors


http://www.qwhale.net
contact@qwhale.net
*/
#endregion Copyright (c) 2004 Quantum Whale LLC.
using
namespace MainDemo
{
public class Test
{
public MainForm()...
/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose(bool disposing
{
this.
if(d
{
Dispose
Main
MainForm_Load
MainForm
void Dispose (bool disposing)
Clean up any resources being used.

Compilers

- Takes program source code as input and produce object code for machine execution
- Program with errors are returned for correction.
- Some compilers rearrange the code for faster object code.

Debuggers

- Program failure is termed as **Bug**
- **Debug**: *to correct the error found in the program*.
- The methods to debug:
  - hand simulate the program
  - Program execution instruction by instruction and observing the results.
  - Tools are developed to inspect the state of the machine during execution.
  - Popular debugging tools are visual debuggers which provide the graphical representation of the programs execution.

Integrated Development Environment (IDE)

- IDEs comprises of
  - Editors
  - Compilers
  - Debuggers
  - Tools for program documentation & maintenance

END of Lecture 9