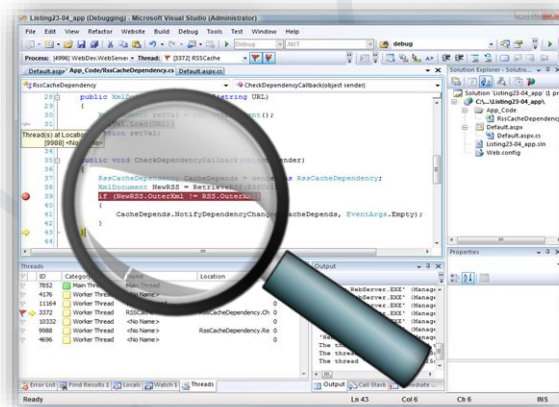


Debugging

Building Rock-Solid Software



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>

sli.do

#csharp-advanced

Table of Contents

1. Introduction to Debugging
2. Visual Studio Debugger
3. Breakpoints
4. Data Inspection
5. Threads and Stacks
6. Finding a Defect





Introduction to Debugging

What is Debugging?

- The process of locating and fixing or bypassing **bugs** (errors) in computer program code
- To **debug** a program:
 - Start with a **problem**
 - Isolate the **source** of the problem
 - **Fix** it
- **Debugging tools** (called **debuggers**) help identify coding errors at various development stages

■ Testing

- A means of initial detection of errors
- The process of verifying and validating that a software or application is bug free

■ Debugging

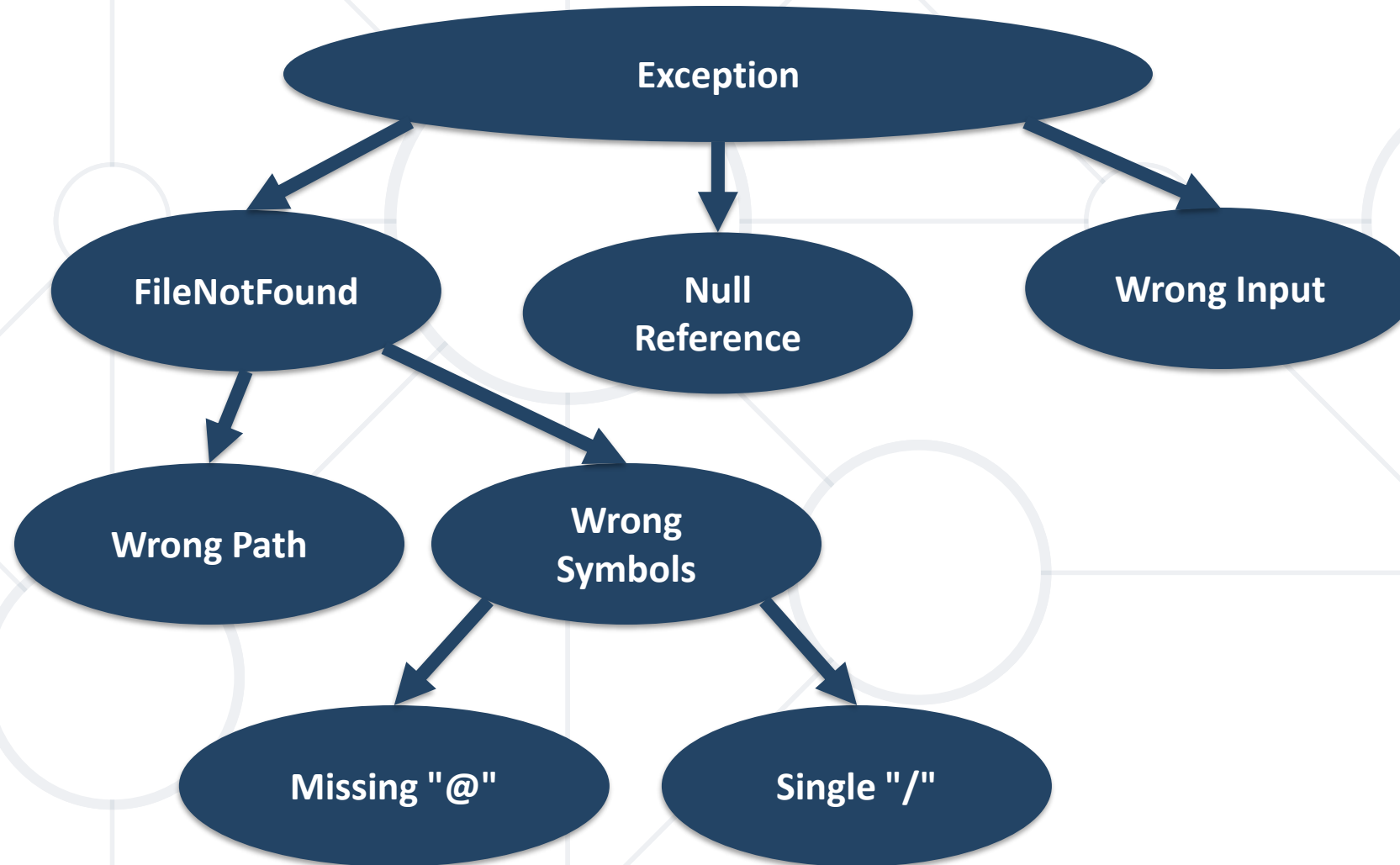
- A means of diagnosing and correcting the root causes of errors that have already been detected
- The process of identifying, analyzing and fixing a bug in the software



- \$60 Billion per year in economic **losses** due to software **defects**
 - E.g. the **Cluster spacecraft failure** was caused by a bug
- Perfect code is an **illusion**
 - There are factors that are out of our control
- **Legacy** code
 - You should be able to debug code that is written years ago
- **Deeper understanding** of system as a whole

- Debugging can be viewed as one big **decision tree**
 - Individual nodes represent **theories**
 - Leaf nodes represent possible **root causes**
 - Traversal of tree boils down to process state **inspection**
 - Minimizing time to resolution is **key**
 - Careful traversal of the decision tree
 - Pattern recognition
 - Visualization and ease of use helps minimize time to resolution

Example Debugging – Decision Tree





Visual Studio Debugger

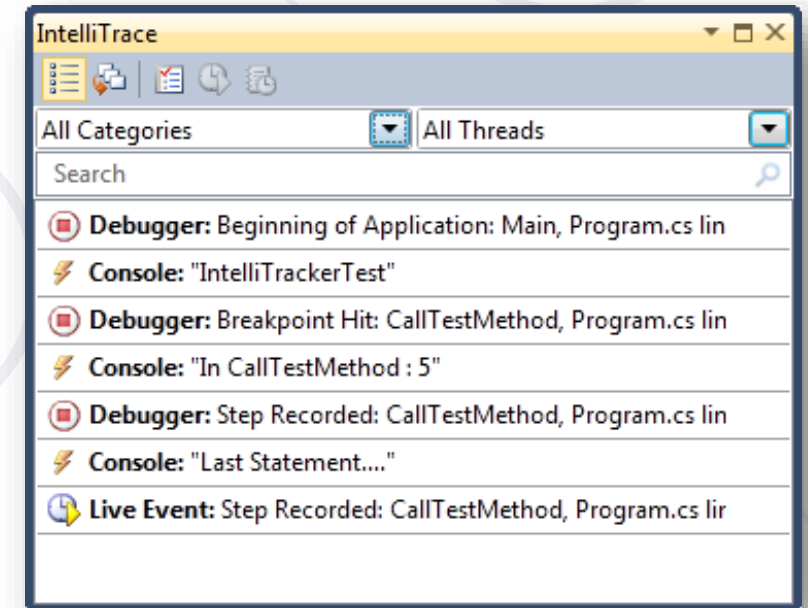
- Visual Studio IDE gives us a lot of **tools** to **debug** your application
 - Adding **breakpoints**
 - Visualize the **program flow**
 - Control the **flow of execution**
 - **Data tips**
 - **Watch variables**
 - Debugging **multithreaded programs**
 - And many more...

- Debug menu, Start Debugging item
 - **F5** is a shortcut
- Easier access to the source code and symbols since its loaded in the solution
- Certain differences exist in comparison to debugging an already running process
 - Hosting for an **ASP.NET application**
 - Visual Studio uses a replacement of the real IIS

- Debug Windows are the means to introspect on the state of a process
- Opens a new window with the selected information in it
- Window categories
 - **Data inspection**
 - **Threading**
- Accessible from menu
 - **Debug -> Windows**

- Convenient shortcut to common debugging tasks
 - **Step into**
 - **Step over**
 - **Continue**
 - **Break**
 - **Breakpoints**
- Customizable to fit your needs
 - **Add / Remove** buttons

- **IntelliTrace** operates in the background, records what you are doing during debugging
- You can easily get a past state of your application from IntelliTrace
- You can **navigate through** your code and see what's happened
- To navigate, just click any of the events that you want to explore



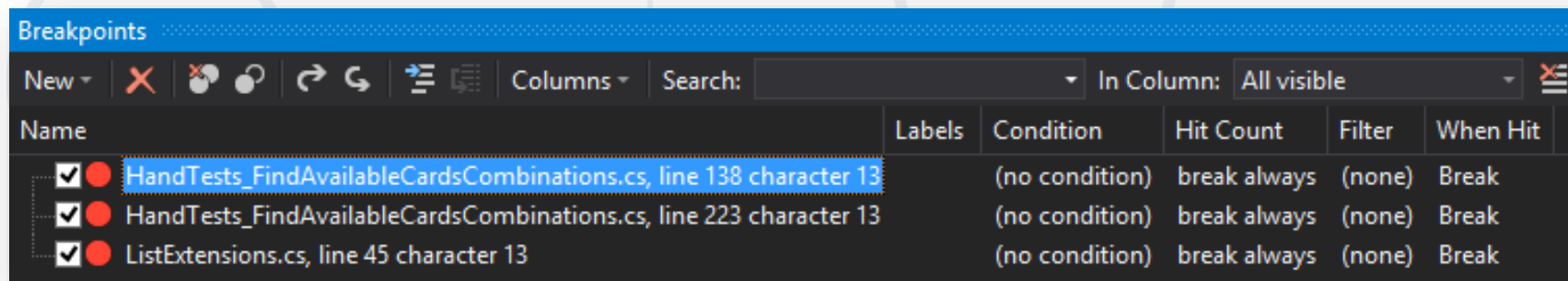


Breakpoints

- Ability to stop execution based on certain criteria is key when debugging
 - When a **function is hit**
 - When **data changes**
 - When a specific **thread** hits a **function**
 - Much more...
- Visual Studio's debugger has a huge feature set when it comes to breakpoints

Managing Breakpoints

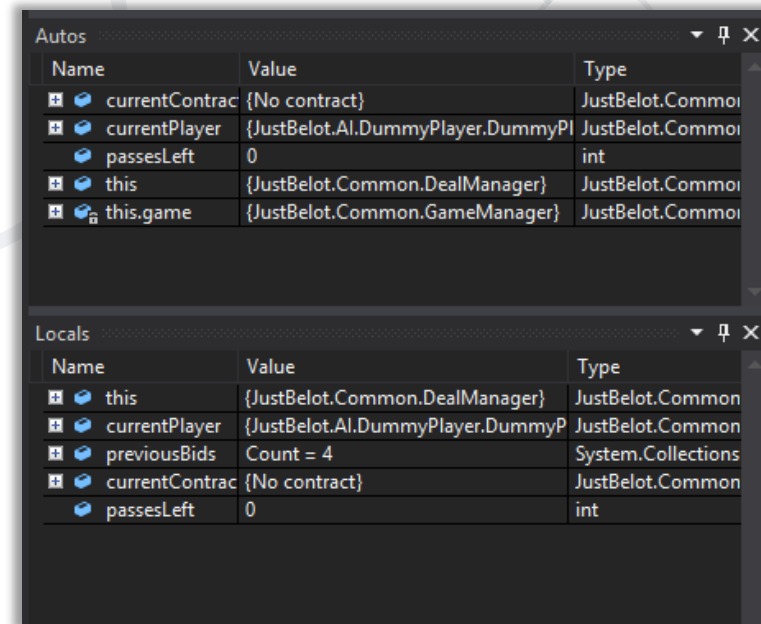
- Managed in the breakpoint window
- Adding breakpoints
- Removing or **disabling** breakpoints
- **Labeling** or **grouping** breakpoints
- Export/import breakpoints





Data Inspection

- Visual Studio offers great data inspection features
 - Watch windows
 - Autos and Locals
 - Memory and Registers
 - Data Tips
 - Immediate window



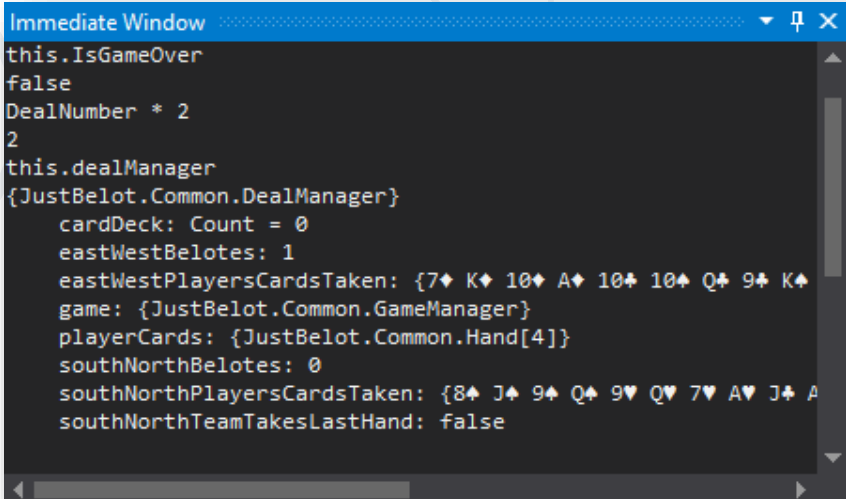
- Allows you to inspect various states of your application
- Several different kinds of "**predefined**" watch windows
 - **Autos**
 - **Locals**
- "**Custom**" watch windows also possible
 - Contains only variables that you choose to add
 - Right click on the variable and select "Add to Watch"

- Locals watch window contains the local variables for the specific stack frame
 - **Debug** -> **Windows** -> **Locals**
 - Displays: name of the variable, value and type
 - Allows drill down into objects by clicking on the + sign in the tree control
- **Autos** lets the debugger decide which variables to show in the window
 - Loosely based on the current and previous statement

- **Memory** window can be used to inspect process wide memory
 - Address field can be a raw pointer or an expression
 - Drag and drop a variable from the source window
 - Number of columns displayed can be configured
 - Data format can be configured
- **Registers** window can be used to inspect processor registers

- Provides information about variables
 - Variables must be within **scope of current execution**
- Place mouse pointer over any variable
 - Variables can be expanded by using the **+ sign**
- Pinning the data tip causes it to **always stay open**
- Comments can be added to data tips
- Data tips support drag and drop
- Importing and exporting data tips

- Useful when debugging due to the **expansive expressions** that can be **executed**
 - To output the value of a variable {**name of variable**}
 - To set values, use {**name of variable**}={**value**}
 - To call a method, use {**name of variable**}.<**method**>(**arguments**)
- Similar to regular code
- Supports **IntelliSense**



```
Immediate Window
this.IsGameOver
false
DealNumber * 2
2
this.dealManager
{JustBelot.Common.DealManager}
  cardDeck: Count = 0
  eastWestBelotes: 1
  eastWestPlayersCardsTaken: {7♦ K♦ 10♦ A♦ 10♣ 10♠ Q♣ 9♣ K♣}
  game: {JustBelot.Common.GameManager}
  playerCards: {JustBelot.Common.Hand[4]}
  southNorthBelotes: 0
  southNorthPlayersCardsTaken: {8♠ J♠ 9♠ Q♠ 9♥ Q♥ 7♥ A♥ J♠ A♥}
  southNorthTeamTakesLastHand: false
```

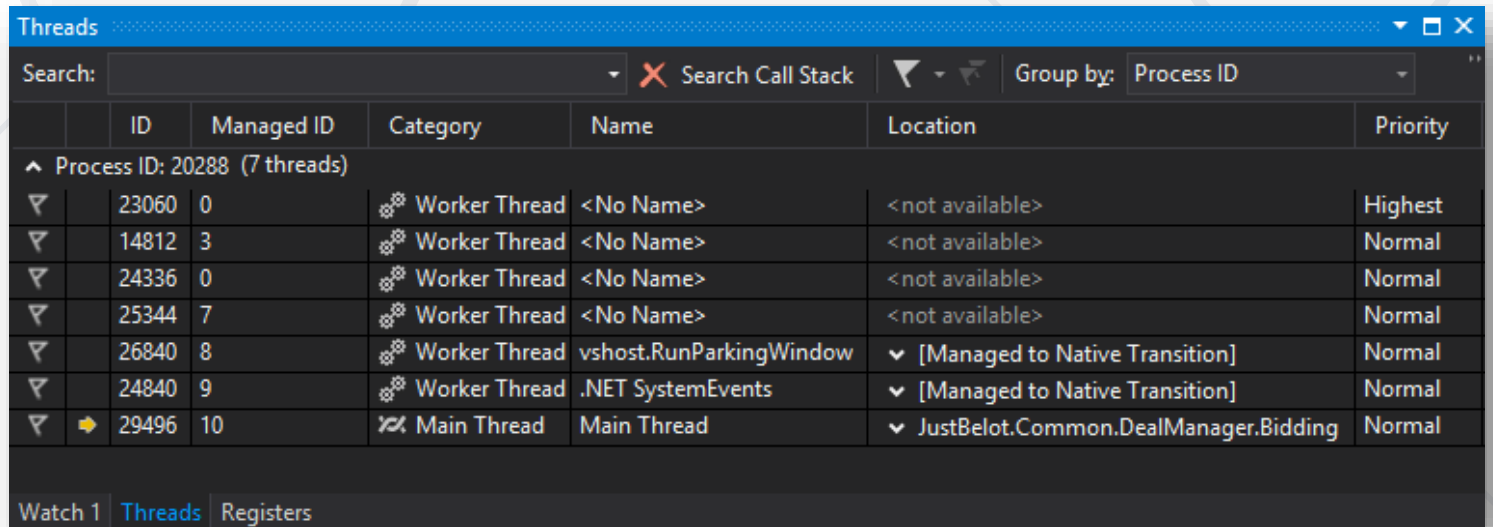


Threads and Stacks

- Fundamental units of code execution
- Commonly, programs use **more** than **one thread**
 - In .NET, there is always more than one thread
- Each thread has a memory area associated with it known as a **stack**
 - Stores **local variables**
 - Stores frame **specific information**
- Memory area employs last in first out semantics

Threads Window

- Contains an overview of thread activity in the process
- Includes basic information in a per thread basis
 - Thread ID's
 - Category
 - Name
 - Location
 - Priority



	ID	Managed ID	Category	Name	Location	Priority
^ Process ID: 20288 (7 threads)						
▼	23060	0	Worker Thread	<No Name>	<not available>	Highest
▼	14812	3	Worker Thread	<No Name>	<not available>	Normal
▼	24336	0	Worker Thread	<No Name>	<not available>	Normal
▼	25344	7	Worker Thread	<No Name>	<not available>	Normal
▼	26840	8	Worker Thread	vshost.RunParkingWindow	▼ [Managed to Native Transition]	Normal
▼	24840	9	Worker Thread	.NET SystemEvents	▼ [Managed to Native Transition]	Normal
▼	29496	10	Main Thread	Main Thread	▼ JustBelot.Common.DealManager.Bidding	Normal

Watch 1 Threads Registers

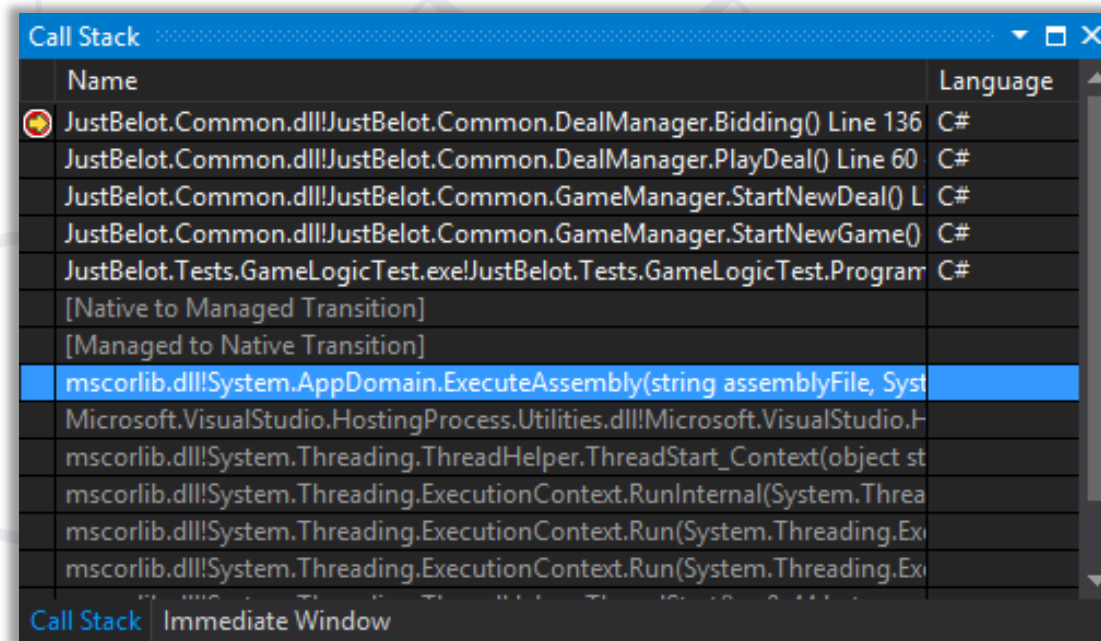
Breakpoint Filters

- Allows you to exert even more control of when a breakpoint hits
- Examples of customization
 - Machine **name**
 - Process **ID**
 - Process **name**
 - Thread **ID**
 - Thread **name**
- Multiple can be combined using &, ||, !



Call Stacks

- Visual Studio shows the elements of a call stack
 - Local variables
 - Method frames





Finding a Defect

Tips for Finding Defects

- Use all available data
- Refine the test cases
- Check unit tests
- Use available tools
- Reproduce the error in several different ways
- Generate more data to generate more hypotheses
- Use the results of negative tests
- Brainstorm for possible hypotheses



Tips for Finding Defects

- Narrow the suspicious region of the code
- Be suspicious of classes and routines that have had defects before
- Check code that's changed recently
- Expand the suspicious region of the code
- Integrate incrementally
- Check for common defects
- Talk to someone else about the problem
- Take a break from the problem



Fixing a Defect

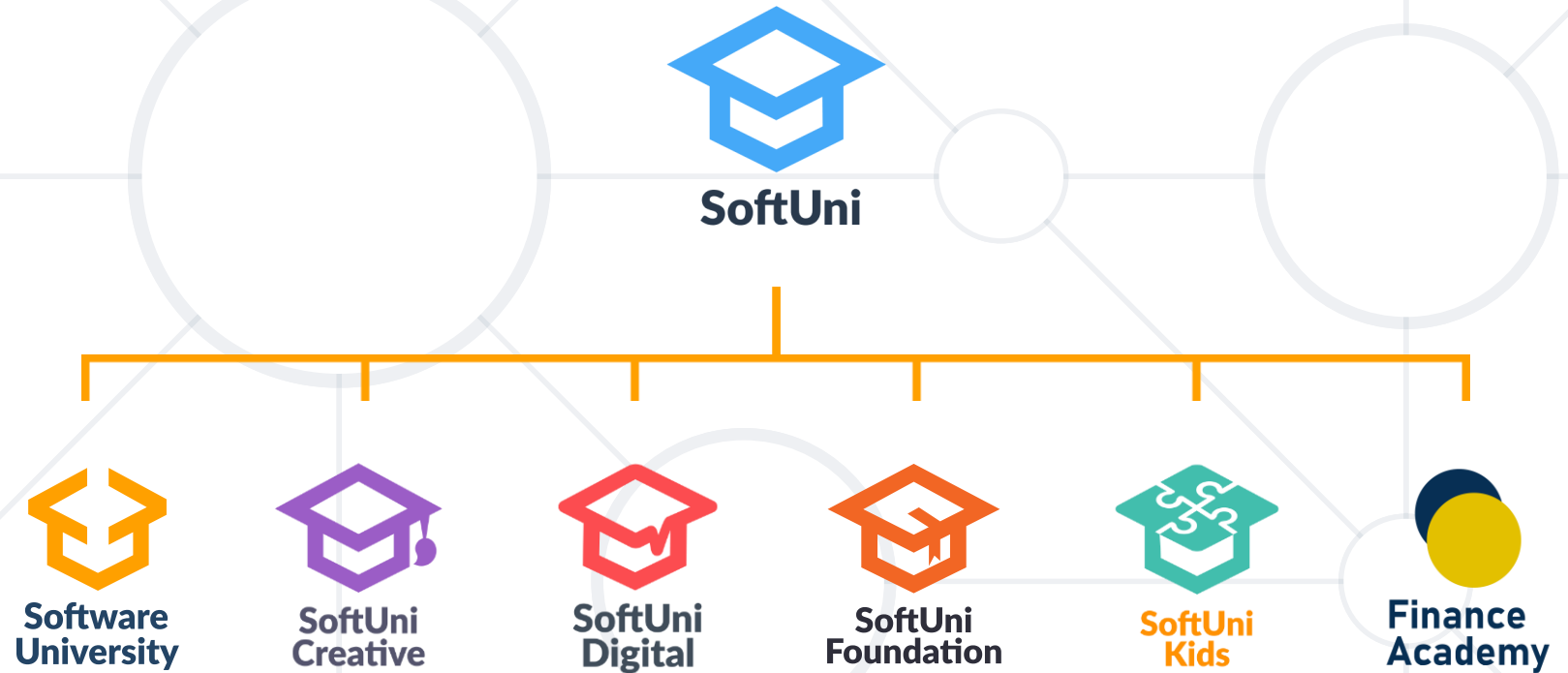
- Understand the problem before you fix it
- Understand the program, not just the problem
- Confirm the defect diagnosis
- Relax
- Save the original source code
- Fix the problem, not the symptom
- Make one change at a time
- Add a unit test that expose the defect
- Look for similar defects



- Introduction to **Debugging**
- Visual Studio Debugger
- **Breakpoints**
- Data Inspection
 - **Locals, Autos, Watch**
- Finding a **Defect**



Questions?



SoftUni Diamond Partners



- Software University – High-Quality Education, Profession and Job for Software Developers

- softuni.bg, about.softuni.bg

- Software University Foundation

- softuni.foundation

- Software University @ Facebook

- facebook.com/SoftwareUniversity



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg>
- © Software University – <https://softuni.bg>

