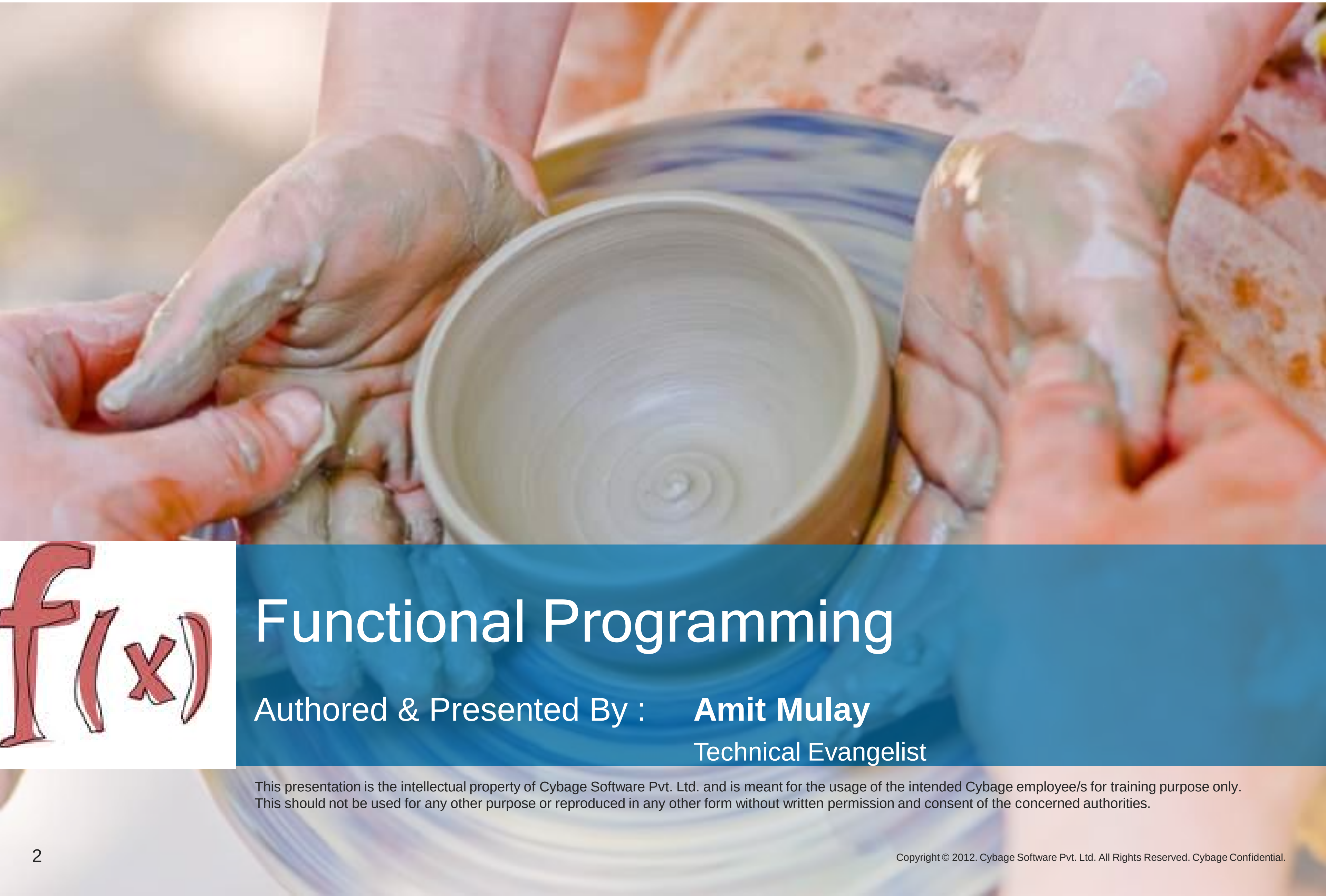
Technology Group

Complexity

Simplified

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.



Functional Programming

Authored & Presented By : **Amit Mulay**

Technical Evangelist

This presentation is the intellectual property of Cybage Software Pvt. Ltd. and is meant for the usage of the intended Cybage employee/s for training purpose only. This should not be used for any other purpose or reproduced in any other form without written permission and consent of the concerned authorities.

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Thinking style

- **Declarative** vs Imperative
- **Compile time** vs Run time
- **Generic** vs non generic

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Declarative programming

- How vs What?
- Statement vs Expression
- Hidden vs explicit

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Declarative vs Imperative

Imperative : Sequence of statements
Declarative : Sequence of expressions

Stone to diamond

Imperative :

1. **Take stone.**
2. **Take hammer.**
3. **Stone is hammered till it becomes diamond**

Declarative : Give me diamond from this stone

Get customer details who are in UK

Imperative : "Take the next customer from a list. If the customer lives in the UK, show their details. If there are more customers in the list, go to the beginning."

Declarative : "Show customer details of every customer living in the UK."

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Maintainability

- Do we understand?
- Global variables
- Pointers to references?
- Action at distance

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Simple Example 1

What will be state of returned ellipse?

```
Ellipse ellipse = new Ellipse(new Rectangle(0, 0, 100, 1
00));
Rectangle boundingBox = ellipse.BoundingBox;
boundingBox.Inflate(10, 10);
return ellipse;
```

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Simple Example 1

```
What will be state of returned ellipse?

Ellipse ellipse = new Ellipse(new Rectangle(0, 0, 100, 1
00));
Rectangle boundingBox = ellipse.BoundingBox;
boundingBox.Inflate(10, 10);
return ellipse;
```

Number of possibilities

1. bounding box could be reference and might inflate ellipse.
2. bounding box could be value type and hence won't have effect on ellipse.
3. Inflate method might have created new rectangle and might have returned that.

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Simple Example 1

```
Solution
Ellipse ellipse = new Ellipse(new Rectangle(0, 0, 100, 1
00));
Rectangle boundingBox = ellipse.BoundingBox;
Rectangle smallerBox = boundingBox.Inflate(10, 10);
return new Ellipse(smallerBox);

Thats immutability!!
```

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Another Example

```
Sequence of lines. (Hidden vs explicit)

var movedMonster = monster.PerformStep();
var inDanger = player.IsCloseTo(movedMonster);

(...)
var hitMonster = monster.HitByShooting(gunShot);
var hitPlayer = player.HitByShooting(gunShot);
(...)
```

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Learnings from two examples

- **What vs How**
- **Productivity**
- **Readability**
- **Reasoning**

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Example 3

- **Requirement**
  - Sum of numbers in a range.

```
int SumNumbers(int from, int to)
{       int res = 0;
        for (int i = from; i <= to; i++)
        res = res + i;
        return res;
}
```

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Example 3

- **Solution**
  - Recursion.

```
int SumNumbers(int from, int to) {
        if (from > to) return 0;
        int sumRest = SumNumbers(from + 1, to);
        return from + sumRest;
}
```

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Example 3

- **First reason**
  - Code is not generic

```
int SumNumbers(int from, int to) {
        if (from > to) return 0;
        int sumRest = SumNumbers(from + 1, to);
        return from + sumRest;
}


int AggregateNumbers(Func<int, int, int> op, int init, i
nt from, int to)
{       if (from > to) return init;
        int sumR = AggregateNumbers(op,init,from+1,to);
        return op(from, sumR);
}
```

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Example 3

- **Solution (F#)**
  - Code is not generic

```
int AggregateNumbers(Func<int, int, int> op, int init, i
nt from, int to)
{       if (from > to) return init;
        int sumR = AggregateNumbers(op,init,from+1,to);
        return op(from, sumR);
}

let rec AggregatNumbers opf init from to =
        if (from > to) return init;
        let sumr = AggregatNumbers op init from+1 to
        return opf(fromr, sumr)
let sumnumbers = AggregatNumbers (+) 0
let multiplynumbers = AggregatNumbers (*) 1
```

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Example 3

- **Second reason**
  - Expression vs statement

```
let rec AggregatNumbers opf init from to =
       if (from > to) return init;
       let sumr = AggregatNumbers op init from+1 to
       return opf(fromr, sumr)
let sumnumbers = AggregatNumbers (+) 0
let multiplynumbers = AggregatNumbers (*) 1
```

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Expression vs Statement

- Statement may or may not return values
- Expression always do return value.
- Statement can modify external data and hence change meaning of next statements which may appear independent
- Expression only works in boundary of expression and hence can't change next part of code which appears independent

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Example 3

- **Not an expression**
  - If else (? : operator)

```
let rec AggregatNumbers op init from to =
       if (from > to) return init;
       let sumr = AggregatNumbers op init from+1 to
       return op(fromr, sumr)

let rec AggregatNumbers op init from to =
       if (from > to) then
        init
       else
        let sumr = AggregatNumbers op init from+1 to
        op(fromr, sumr)
```

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Static vs Dynamic

- **Compile time vs Run time**
- **Generic Sort**
  - **Interface for callback which compares items**
  - **Collection**

```
public class CustomerComparer : IComparer
{
        int IComparer.Compare( Object x, Object y )
        {
                Customer x1 = (Customer) x; // chance of exception
                Customer y1 = (Customer) y; // chance of exception
                x1.name.Compare(y2.name)
        }
 }
```

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Static vs Dynamic

- **Compile time vs Run time**
- **Generic Sort**
  - **Interface for callback which compares items**
  - **Collection**

```
let sort cmpf coll =
        (…)
        match compf ele1 ele2
        | -1 -> (..)
        | 1 -> (..)
        | 0 -> (..)
// compile time check
let sorted = sort (fun (x : Customer) (y : Customer) ->
                      x.name = y.name) coll
```

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Static vs Dynamic

- **Cast exceptions are avoided by relying of compile type checking**
- **There are one more class of exceptions**
  - **Null pointer exception**

This are avoided by

1. No references. All value types.

2. What about representing if something is there or not?

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Static vs Dynamic

- Solutions for null check is use of options

```
Let a = Some x
Let b = None
```

Now whenever you use a and b, Compiler will force you to check and have “expression” for both Something or Nothing conditions.

This is done by pattern matching

```
Match a with
| None -> (none handling)
| Some x -> (Some handling)
```

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Summary

- References are todays global variables
- Value types are preferred over references
- Immutability is preferred
- Declarative coding style is preferred
- Generic coding is preferred
- Compile time checks are preferred.

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Summary

Now, Do we understand?

Imperative : Sequence of statements
Declarative : Sequence of expressions

Stone to diamond

Imperative :

1. **Take stone.**
2. **Take hammer.**
3. **Stone is hammered till it becomes diamond**

Declarative : Give me diamond from this stone

Get customer details who are in UK

Imperative : "Take the next customer from a list. If the customer lives in the UK, show their details. If there are more customers in the list, go to the beginning."

Declarative : "Show customer details of every customer living in the UK."

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Any Questions?

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.

Thank you!

Copyright © 2012. Cybage Software Pvt. Ltd. All Rights Reserved. Cybage Confidential.