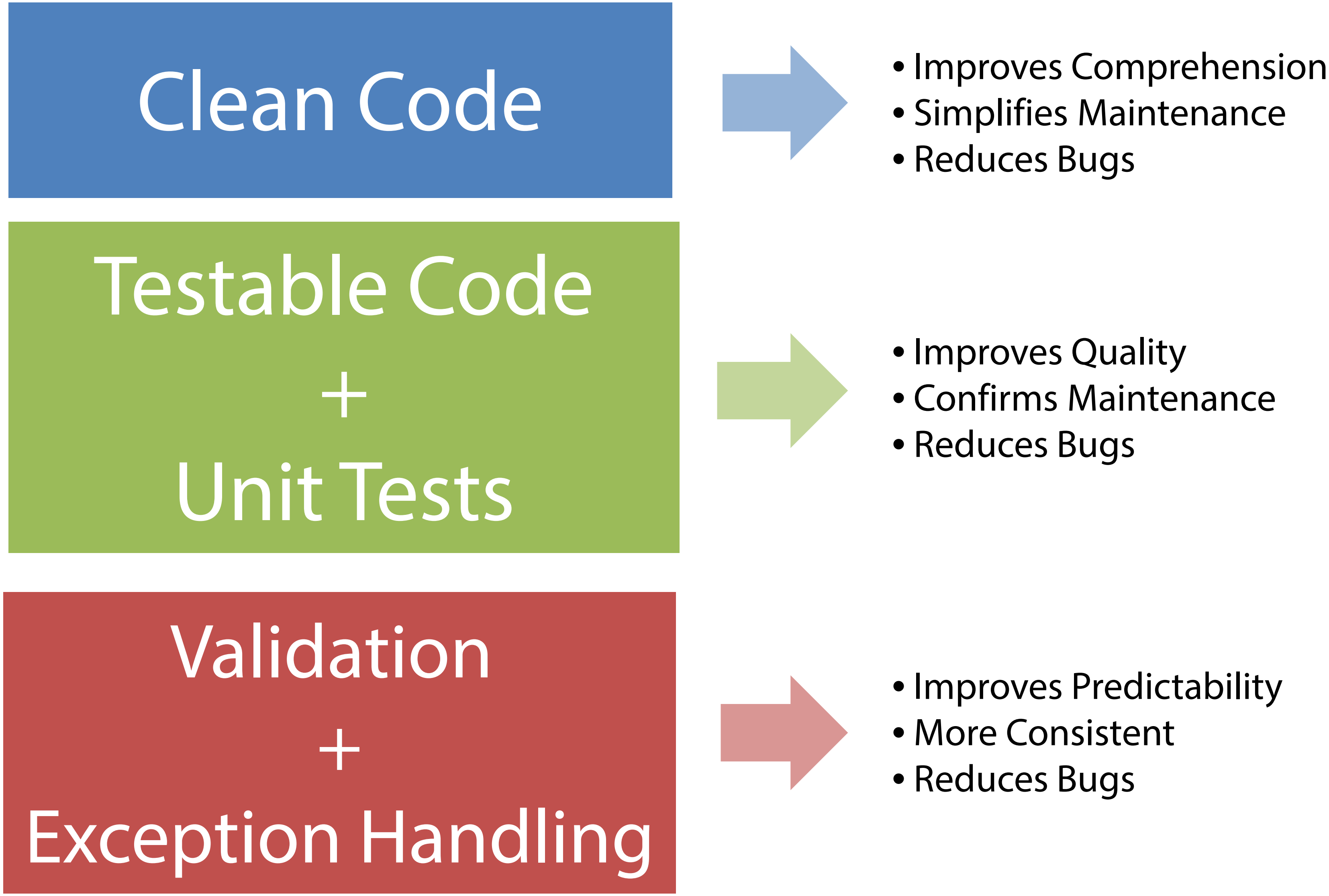
Defending Your Methods

Deborah Kurata
http://msmvps.com/blogs/deborahk/
@DeborahKurata
deborahk@insteptech.com

pluralsight
hardcore dev and IT training

Defensive Coding



Clean Code

- Improves Comprehension
- Simplifies Maintenance
- Reduces Bugs

Testable Code + Unit Tests

- Improves Quality
- Confirms Maintenance
- Reduces Bugs

Validation + Exception Handling

- Improves Predictability
- More Consistent
- Reduces Bugs

Defensive Coding

Clean Code

- Improves Comprehension
- Simplifies Maintenance
- Reduces Bugs

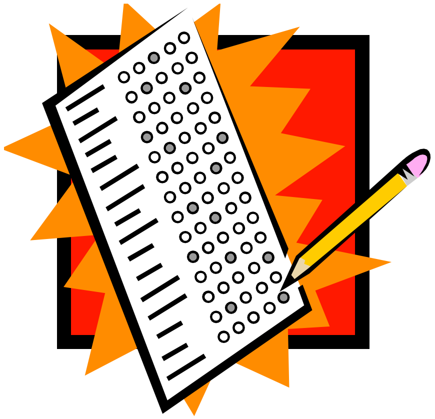
Defensive Coding

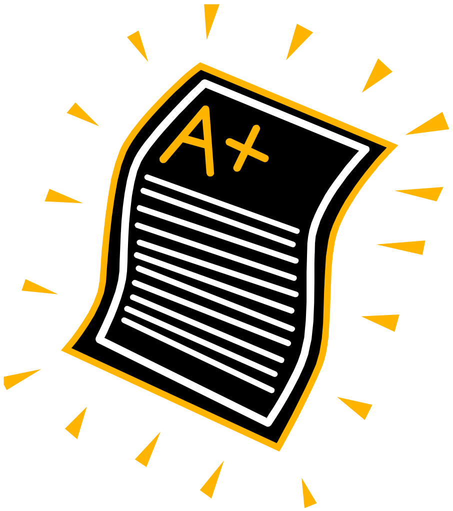
Testable Code

+

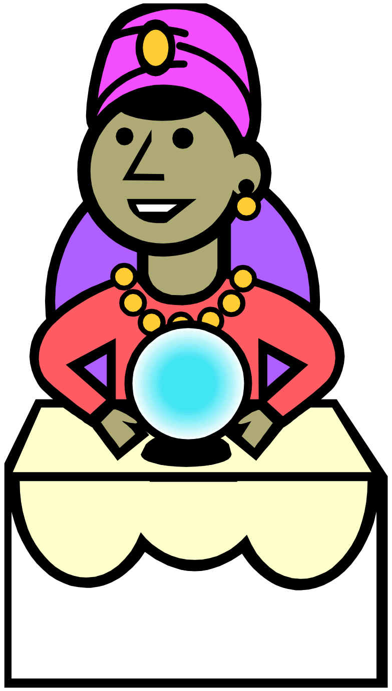
Unit Tests

- Improves Quality
- Validates Maintenance
- Reduces Bugs





Defensive Coding



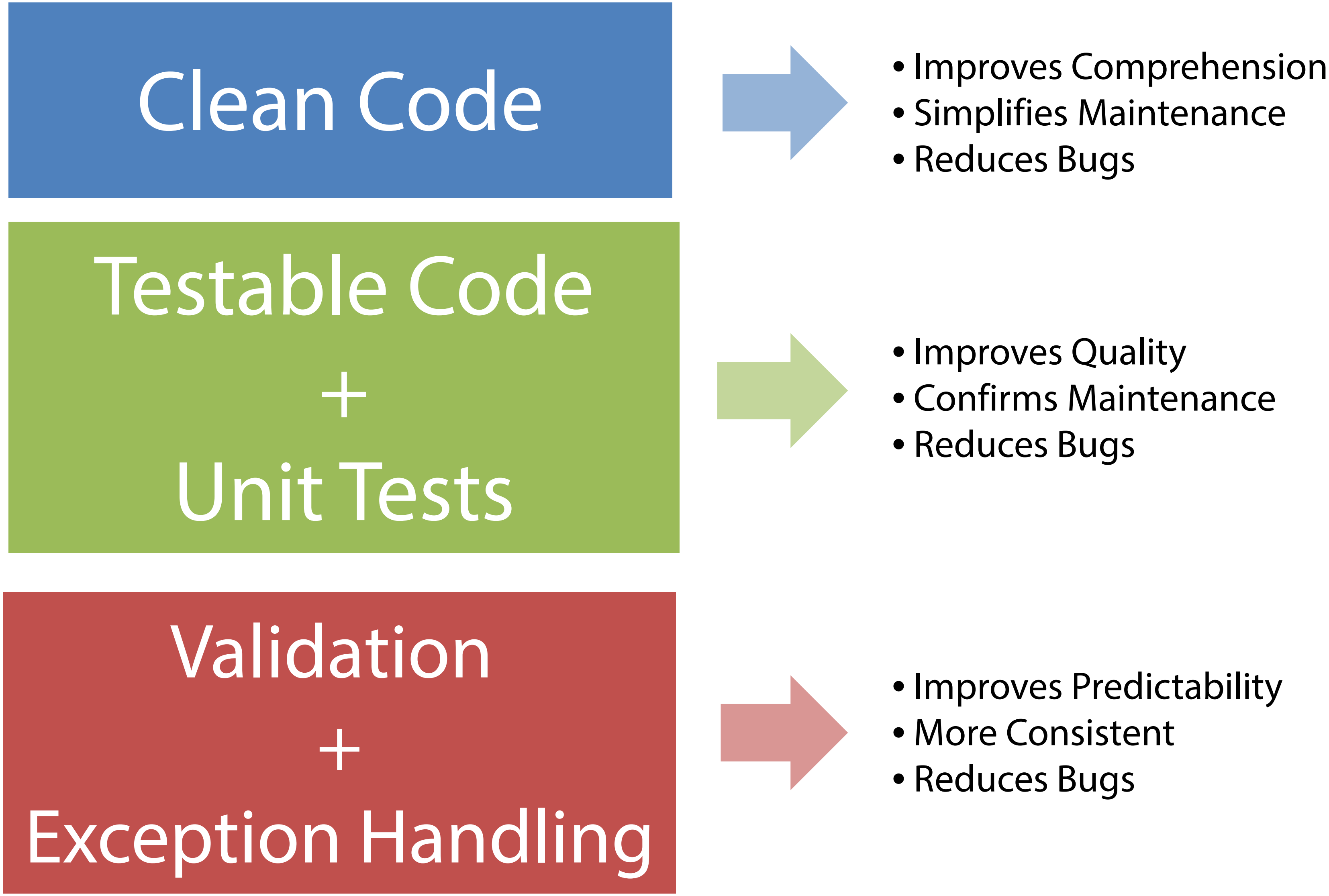
Validation

+

Exception Handling

- Improves Predictability
- More Consistent
- Reduces Bugs

Defensive Coding



Clean Code

- Improves Comprehension
- Simplifies Maintenance
- Reduces Bugs

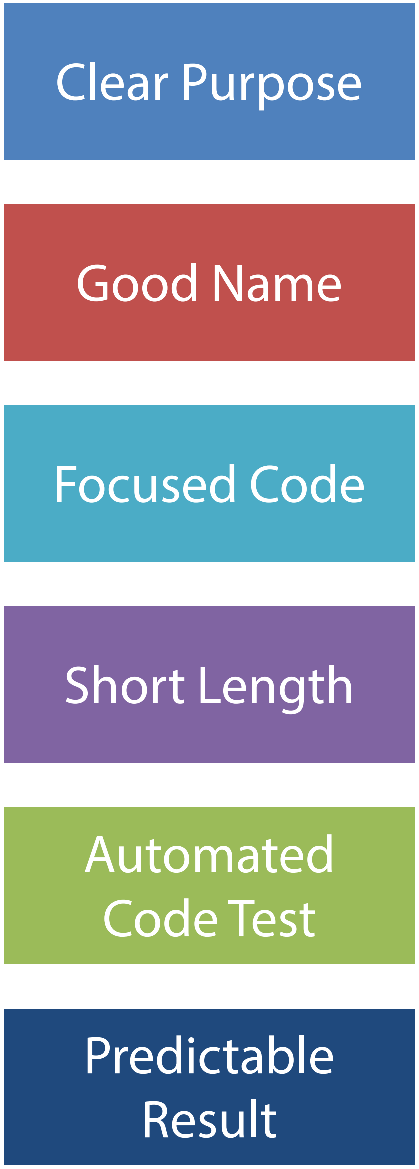
Testable Code
+
Unit Tests

- Improves Quality
- Confirms Maintenance
- Reduces Bugs

Validation
+
Exception Handling

- Improves Predictability
- More Consistent
- Reduces Bugs

Clean, Testable, Predictable Methods



Clear Purpose

Good Name

Focused Code

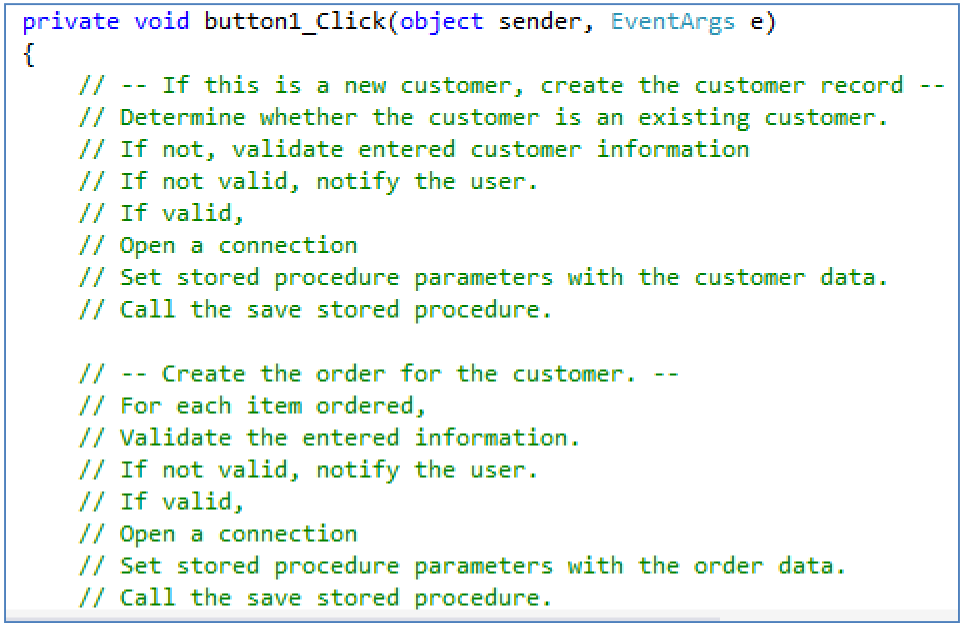
Short Length

Automated Code Test

Predictable Result

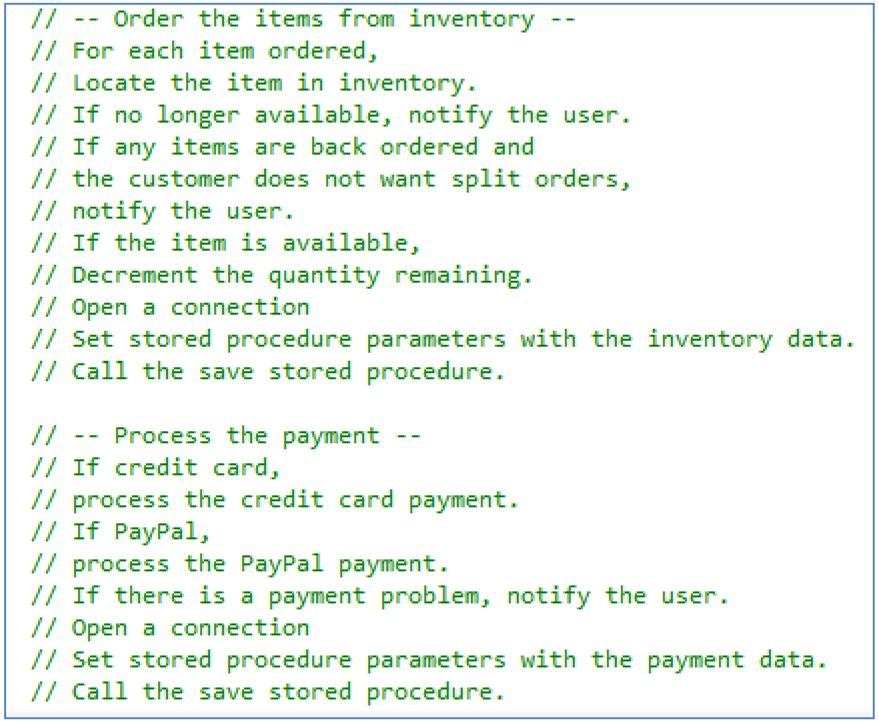
Example

Optionally,
PAUSE NOW



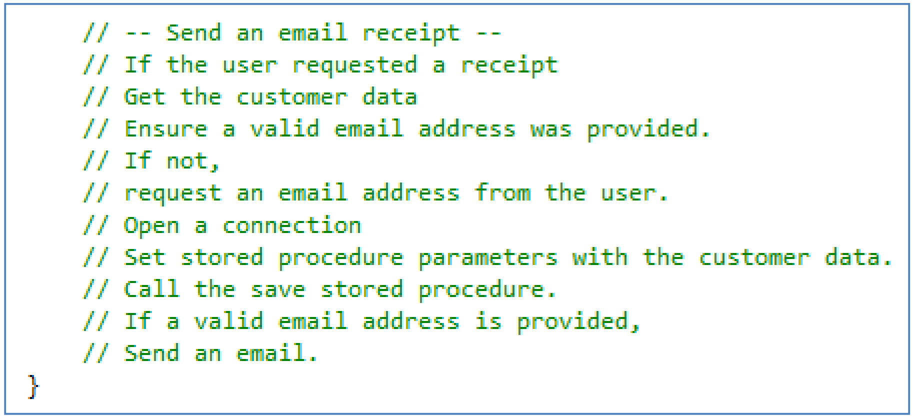
```
private void button1_Click(object sender, EventArgs e)
{
    // -- If this is a new customer, create the customer record --
    // Determine whether the customer is an existing customer.
    // If not, validate entered customer information
    // If not valid, notify the user.
    // If valid,
    // Open a connection
    // Set stored procedure parameters with the customer data.
    // Call the save stored procedure.

    // -- Create the order for the customer. --
    // For each item ordered,
    // Validate the entered information.
    // If not valid, notify the user.
    // If valid,
    // Open a connection
    // Set stored procedure parameters with the order data.
    // Call the save stored procedure.
```



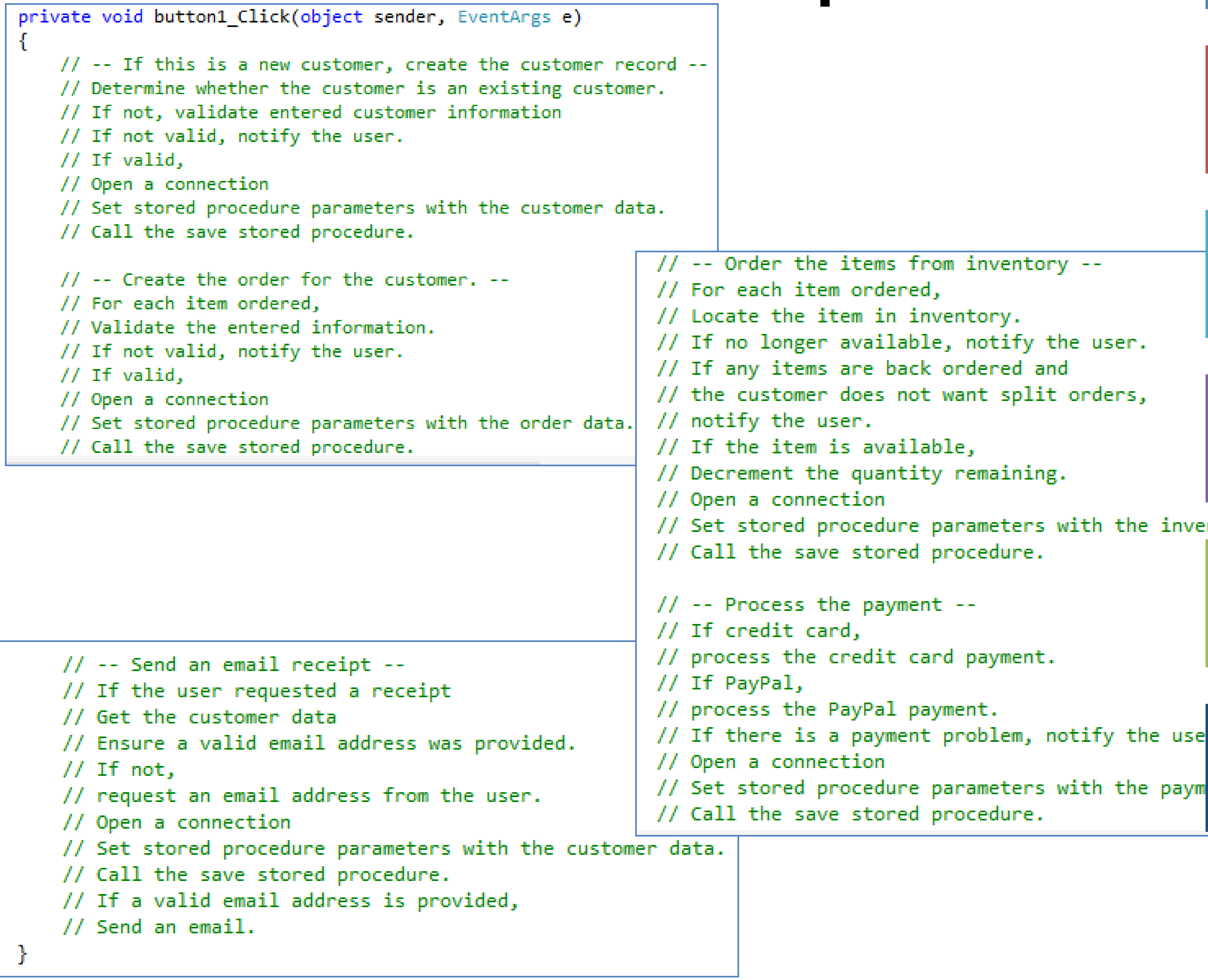
```
// -- Order the items from inventory --
// For each item ordered,
// Locate the item in inventory.
// If no longer available, notify the user.
// If any items are back ordered and
// the customer does not want split orders,
// notify the user.
// If the item is available,
// Decrement the quantity remaining.
// Open a connection
// Set stored procedure parameters with the inventory data.
// Call the save stored procedure.

// -- Process the payment --
// If credit card,
// process the credit card payment.
// If PayPal,
// process the PayPal payment.
// If there is a payment problem, notify the user.
// Open a connection
// Set stored procedure parameters with the payment data.
// Call the save stored procedure.
```



```
    // -- Send an email receipt --
    // If the user requested a receipt
    // Get the customer data
    // Ensure a valid email address was provided.
    // If not,
    // request an email address from the user.
    // Open a connection
    // Set stored procedure parameters with the customer data.
    // Call the save stored procedure.
    // If a valid email address is provided,
    // Send an email.
}
```

Example



```
private void button1_Click(object sender, EventArgs e)
{
    // -- If this is a new customer, create the customer record --
    // Determine whether the customer is an existing customer.
    // If not, validate entered customer information
    // If not valid, notify the user.
    // If valid,
    // Open a connection
    // Set stored procedure parameters with the customer data.
    // Call the save stored procedure.

    // -- Create the order for the customer. --
    // For each item ordered,
    // Validate the entered information.
    // If not valid, notify the user.
    // If valid,
    // Open a connection
    // Set stored procedure parameters with the order data.
    // Call the save stored procedure.

    // -- Order the items from inventory --
    // For each item ordered,
    // Locate the item in inventory.
    // If no longer available, notify the user.
    // If any items are back ordered and
    // the customer does not want split orders,
    // notify the user.
    // If the item is available,
    // Decrement the quantity remaining.
    // Open a connection
    // Set stored procedure parameters with the inventory data.
    // Call the save stored procedure.

    // -- Process the payment --
    // If credit card,
    // process the credit card payment.
    // If PayPal,
    // process the PayPal payment.
    // If there is a payment problem, notify the use
    // Open a connection
    // Set stored procedure parameters with the paym
    // Call the save stored procedure.

    // -- Send an email receipt --
    // If the user requested a receipt
    // Get the customer data
    // Ensure a valid email address was provided.
    // If not,
    // request an email address from the user.
    // Open a connection
    // Set stored procedure parameters with the customer data.
    // Call the save stored procedure.
    // If a valid email address is provided,
    // Send an email.
}
```

Clear Purpose

Good Name

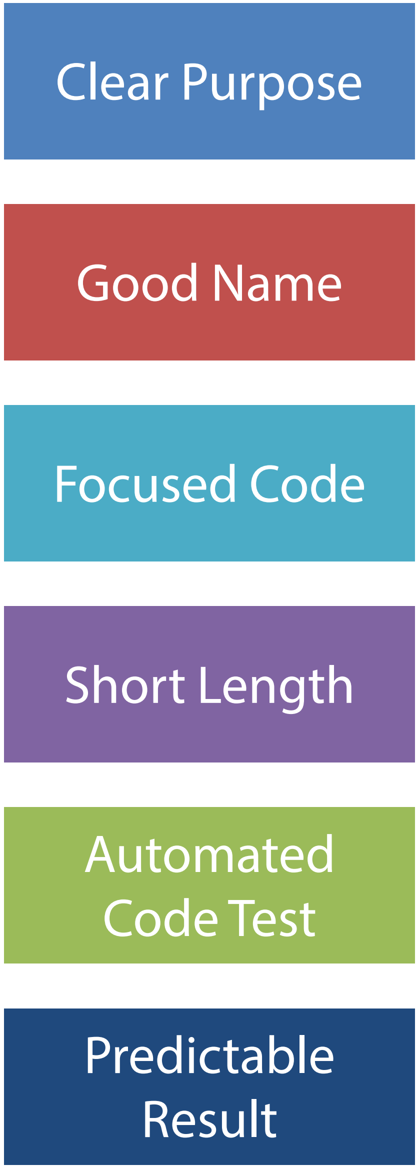
Focused Code

Short Length

Automated Code Test

Predictable Result

Clean, Testable, and Predictable Methods



Clear Purpose

Good Name

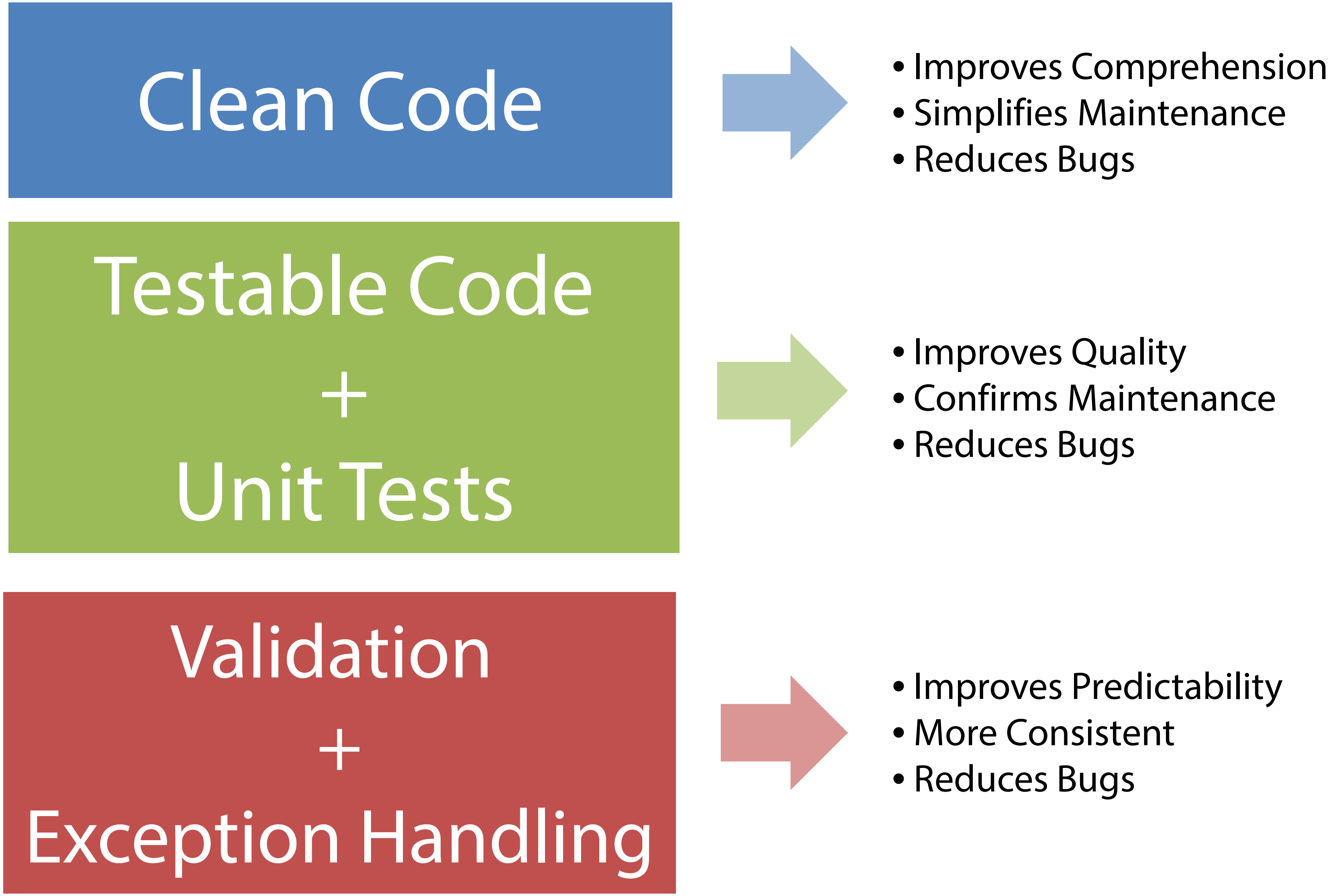
Focused Code

Short Length

Automated Code Test

Predictable Result

Defensive Coding



Clean Code

- Improves Comprehension
- Simplifies Maintenance
- Reduces Bugs

Testable Code
+
Unit Tests

- Improves Quality
- Confirms Maintenance
- Reduces Bugs

Validation
+
Exception Handling

- Improves Predictability
- More Consistent
- Reduces Bugs