ORACLE®

OSvC TECHNICAL SUPPORT WEBINARS:

Demystifying CPMs

our experts are your guides

Johnny Meehan
Senior Technical Support Engineer
OSvC Technical Support

June 6, 2018

ORACLE

Safe Harbor Statement

Agenda

1. CPM Overview
2. Code Walkthrough
3. Debugging
4. Advanced Notions
5. Q&A

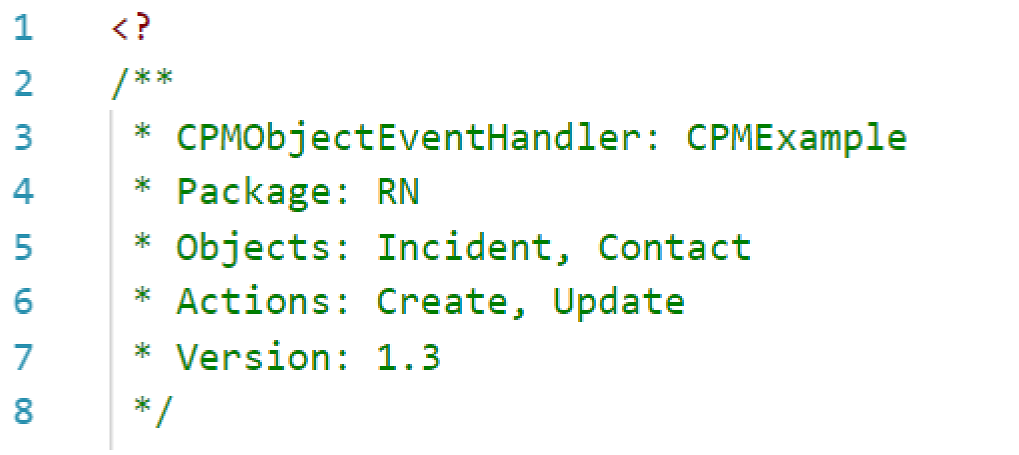
CPM Overview

What are Custom Process Models (CPMs)?

- Server-side business logic (a.k.a. PHP scripts)
- Can be triggered by Create, Update, Delete operations or Business Rules
- Typical uses include:
  - Syncing data between systems (i.e. update contact master records)
  - Calculating and recording detailed metrics
  - Sending notifications

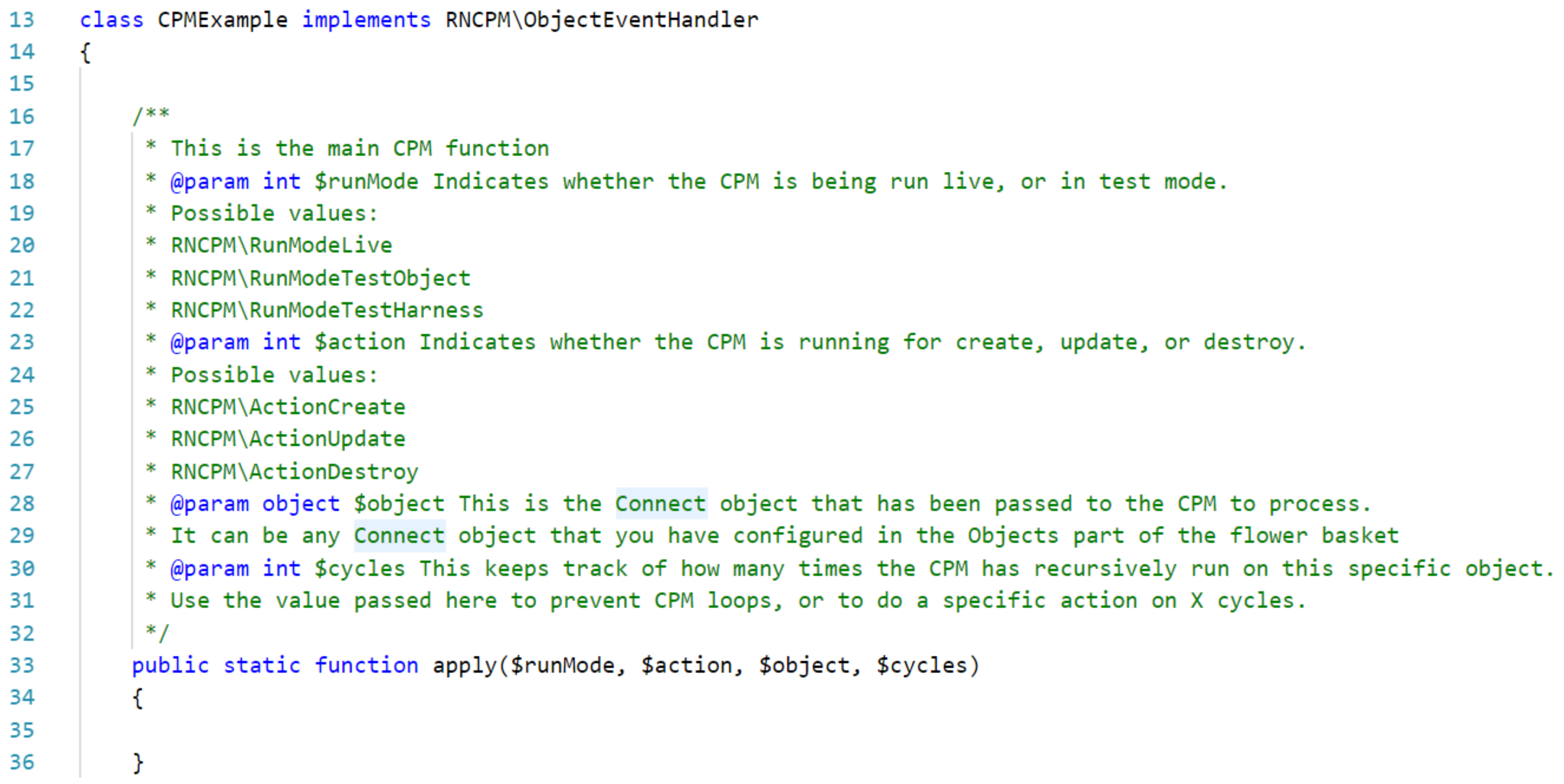
Code Walkthrough

Flower Basket



```
<?
/**
 * CPMObjectEventHandler: CPMExample
 * Package: RN
 * Objects: Incident, Contact
 * Actions: Create, Update
 * Version: 1.3
 */
```

Main Class & Apply Function

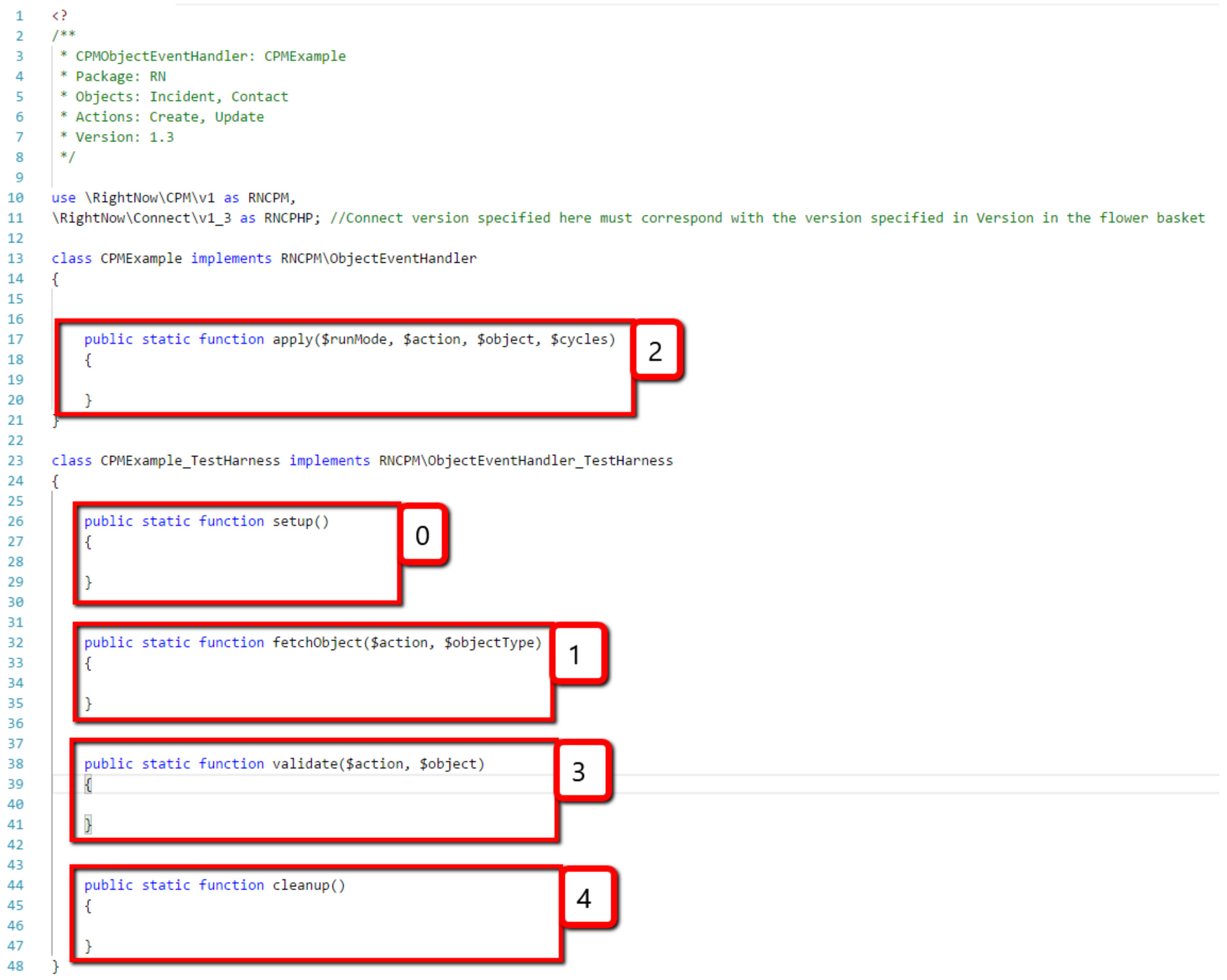


```
class CPMExample implements RNCPM\ObjectEventHandler
{

    /**
     * This is the main CPM function
     * @param int $runMode Indicates whether the CPM is being run live, or in test mode.
     * Possible values:
     * RNCPM\RunModeLive
     * RNCPM\RunModeTestObject
     * RNCPM\RunModeTestHarness
     * @param int $action Indicates whether the CPM is running for create, update, or destroy.
     * Possible values:
     * RNCPM\ActionCreate
     * RNCPM\ActionUpdate
     * RNCPM\ActionDestroy
     * @param object $object This is the Connect object that has been passed to the CPM to process.
     * It can be any Connect object that you have configured in the Objects part of the flower basket
     * @param int $cycles This keeps track of how many times the CPM has recursively run on this specific object.
     * Use the value passed here to prevent CPM loops, or to do a specific action on X cycles.
     */
    public static function apply($runMode, $action, $object, $cycles)
    {

    }
```

Test Harness

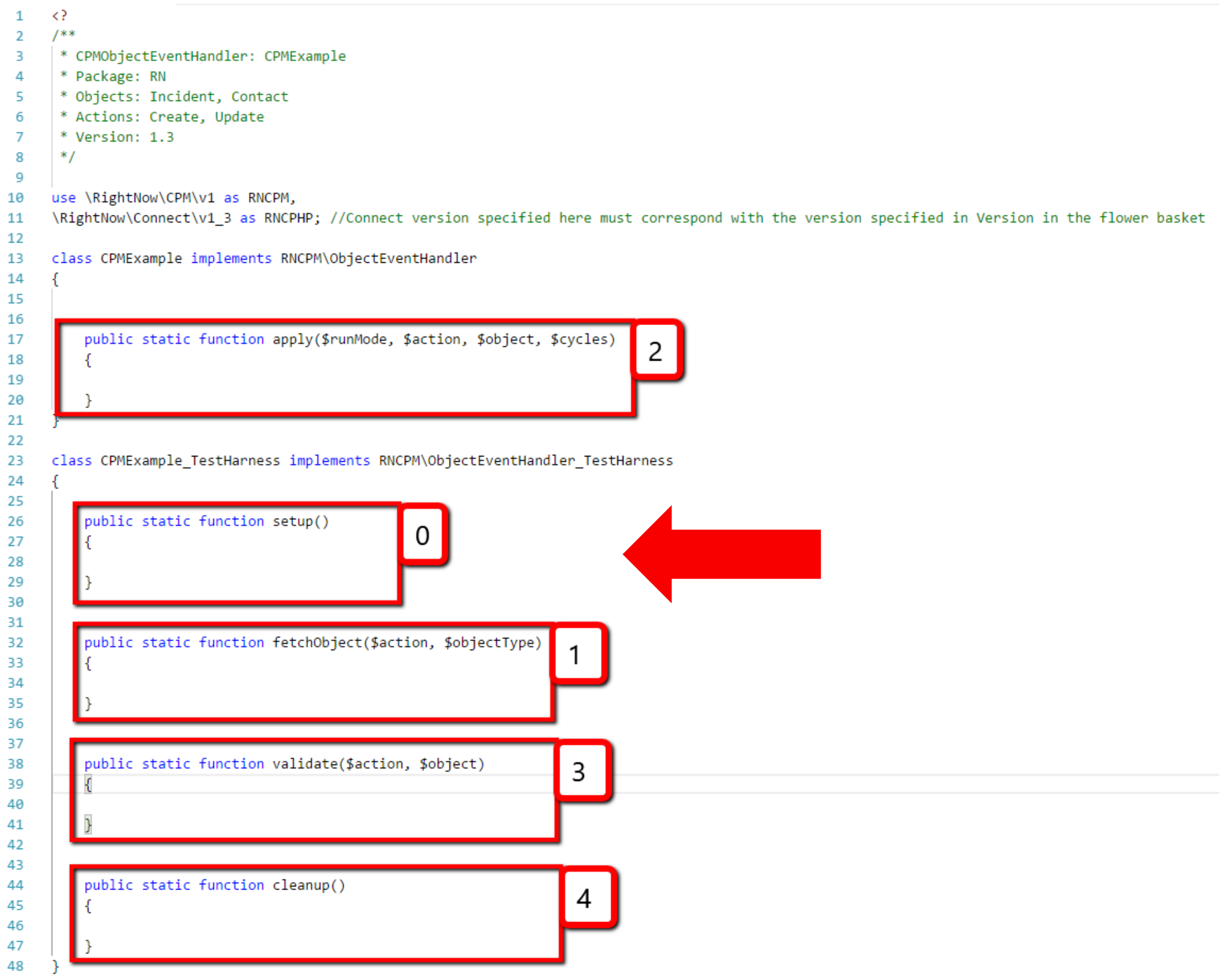
```php
class CPMExample_TestHarness implements RNCPM\ObjectEventHandler_TestHarness
{

    public static function setup()
    {

    }


    public static function fetchObject($action, $objectType)
    {

    }


    public static function validate($action, $object)
    {

    }


    public static function cleanup()
    {

    }
}
```

Test Harness Order of Operations



```php
<?
/**
 * CPMObjectEventHandler: CPMExample
 * Package: RN
 * Objects: Incident, Contact
 * Actions: Create, Update
 * Version: 1.3
 */

use \RightNow\CPM\v1 as RNCPM,
\RightNow\Connect\v1_3 as RNCPHP; //Connect version specified here must correspond with the version specified in Version in the flower basket

class CPMExample implements RNCPM\ObjectEventHandler
{


    public static function apply($runMode, $action, $object, $cycles)
    {

    }
}

class CPMExample_TestHarness implements RNCPM\ObjectEventHandler_TestHarness
{

    public static function setup()
    {

    }

    public static function fetchObject($action, $objectType)
    {

    }

    public static function validate($action, $object)
    {

    }

    public static function cleanup()
    {

    }
}
```

Function: setup()



```
<?
/**
 * CPMObjectEventHandler: CPMExample
 * Package: RN
 * Objects: Incident, Contact
 * Actions: Create, Update
 * Version: 1.3
 */

use \RightNow\CPM\v1 as RNCPM,
\RightNow\Connect\v1_3 as RNCPHP; //Connect version specified here must correspond with the version specified in Version in the flower basket

class CPMExample implements RNCPM\ObjectEventHandler
{


    public static function apply($runMode, $action, $object, $cycles)
    {

    }
}

class CPMExample_TestHarness implements RNCPM\ObjectEventHandler_TestHarness
{

    public static function setup()
    {

    }

    public static function fetchObject($action, $objectType)
    {

    }

    public static function validate($action, $object)
    {

    }

    public static function cleanup()
    {

    }
}
```

Function: setup()

```php
private static $testIncident;
private static $testContact;
/**
 * This function runs once per test, before any of the other functions.
 * It is typically used to create test objects, or any other one-time-only setup.
 */
public static function setup()
{
    $contact = new RNCPHP\Contact();
    $contact->Name = new RNCPHP\PersonName();
    $contact->Name->First = "Malcolm";
    $contact->Name->Last = "Reynolds";
    $contact->save();

    static::$testContact = $contact;

    $incident = new RNCPHP\Incident();
    $incident->PrimaryContact = $contact;
    $incident->Subject = "Test Incident";

    static::$testIncident = $incident;
}
```

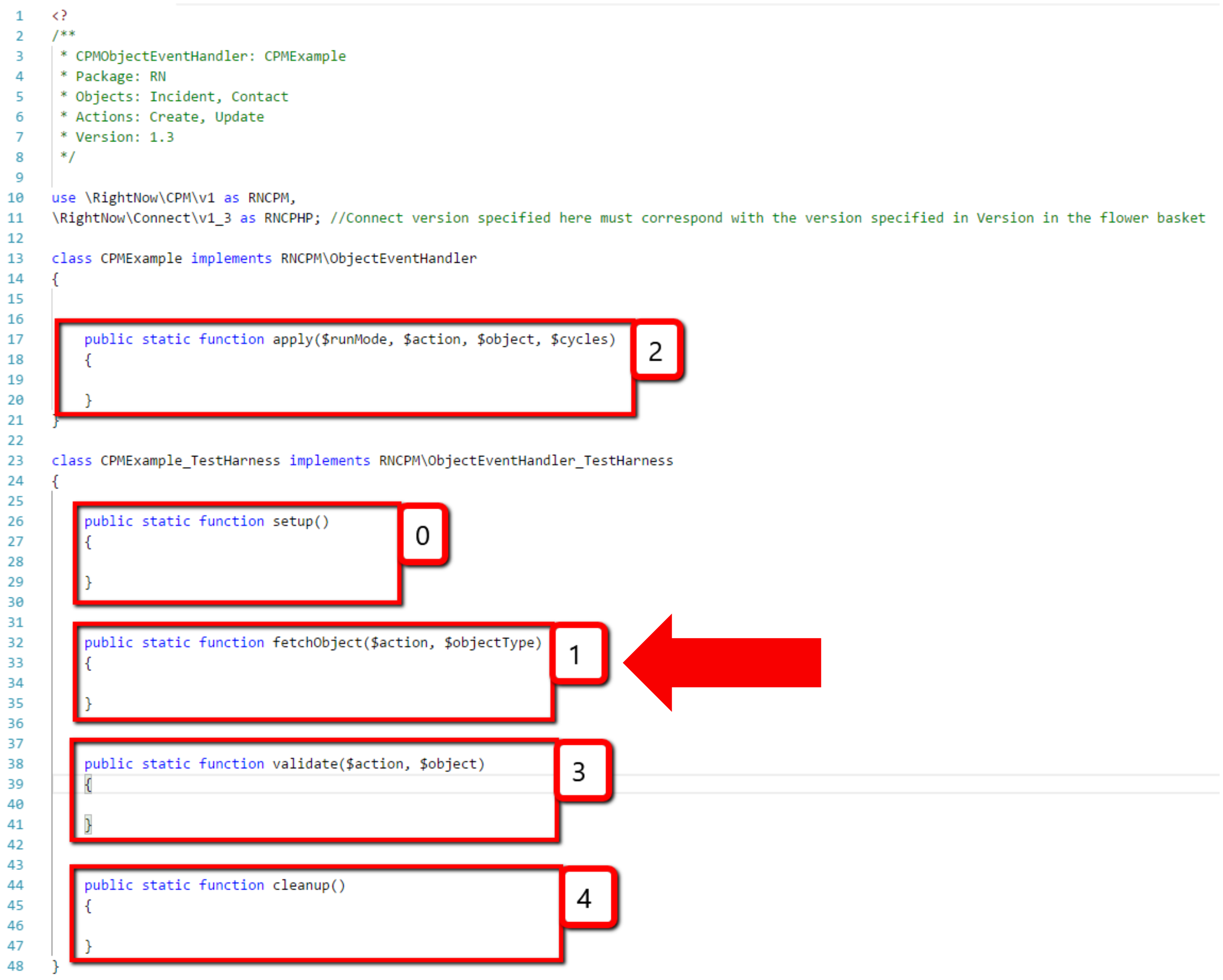
Function: setup()

```
private static $testIncident;
private static $testContact;
/**
 * This function runs once per test, before any of the other functions.
 * It is typically used to create test objects, or any other one-time-only setup.
 */
public static function setup()
{
    $contact = new RNCPHP\Contact();
    $contact->Name = new RNCPHP\PersonName();
    $contact->Name->First = "Malcolm";
    $contact->Name->Last = "Reynolds";
    $contact->save();

    static::$testContact = $contact;

    $incident = new RNCPHP\Incident();
    $incident->PrimaryContact = $contact;
    $incident->Subject = "Test Incident";

    static::$testIncident = $incident;
}
```

Function: setup()

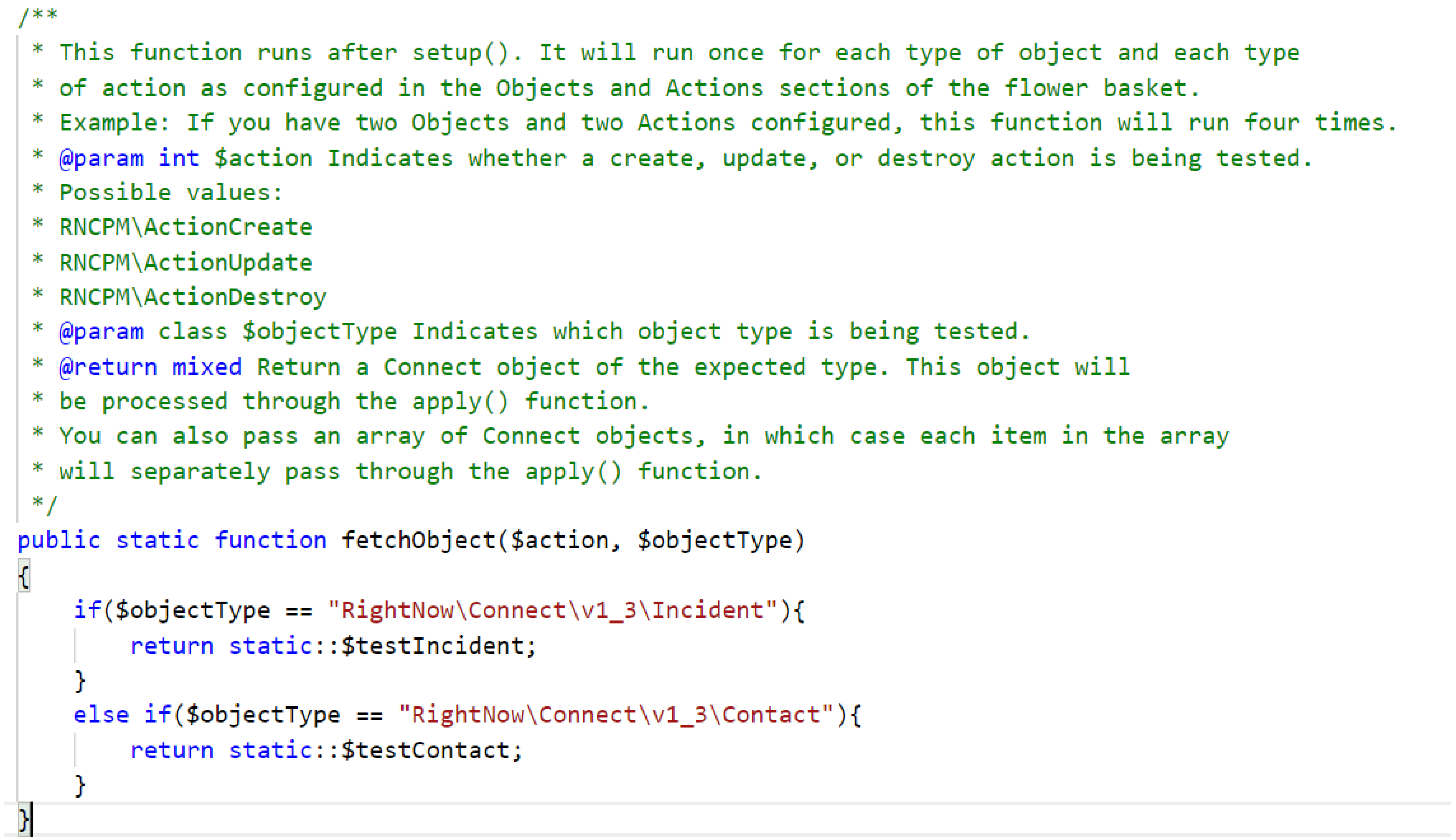
```php
private static $testIncident;
private static $testContact;
/**
 * This function runs once per test, before any of the other functions.
 * It is typically used to create test objects, or any other one-time-only setup.
 */
public static function setup()
{
    $contact = new RNCPHP\Contact();
    $contact->Name = new RNCPHP\PersonName();
    $contact->Name->First = "Malcolm";
    $contact->Name->Last = "Reynolds";
    $contact->save();

    static::$testContact = $contact;

    $incident = new RNCPHP\Incident();
    $incident->PrimaryContact = $contact;
    $incident->Subject = "Test Incident";

    static::$testIncident = $incident;
}
```

Function: fetchObject()



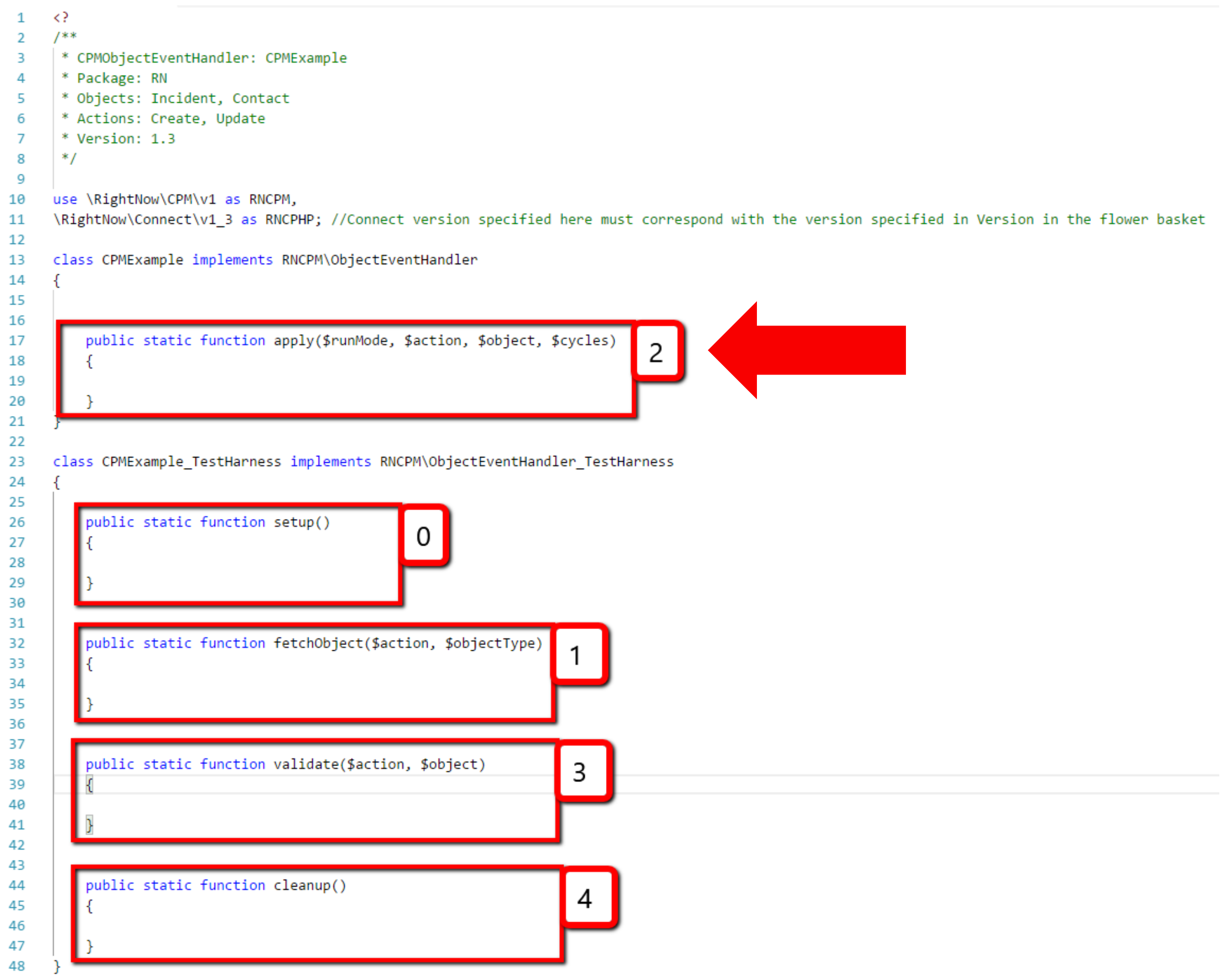
```
<?
/**
 * CPMObjectEventHandler: CPMExample
 * Package: RN
 * Objects: Incident, Contact
 * Actions: Create, Update
 * Version: 1.3
 */

use \RightNow\CPM\v1 as RNCPM,
\RightNow\Connect\v1_3 as RNCPHP; //Connect version specified here must correspond with the version specified in Version in the flower basket

class CPMExample implements RNCPM\ObjectEventHandler
{


    public static function apply($runMode, $action, $object, $cycles)
    {

    }
}

class CPMExample_TestHarness implements RNCPM\ObjectEventHandler_TestHarness
{

    public static function setup()
    {

    }


    public static function fetchObject($action, $objectType)
    {

    }


    public static function validate($action, $object)
    {

    }


    public static function cleanup()
    {

    }
}
```

Function: fetchObject()



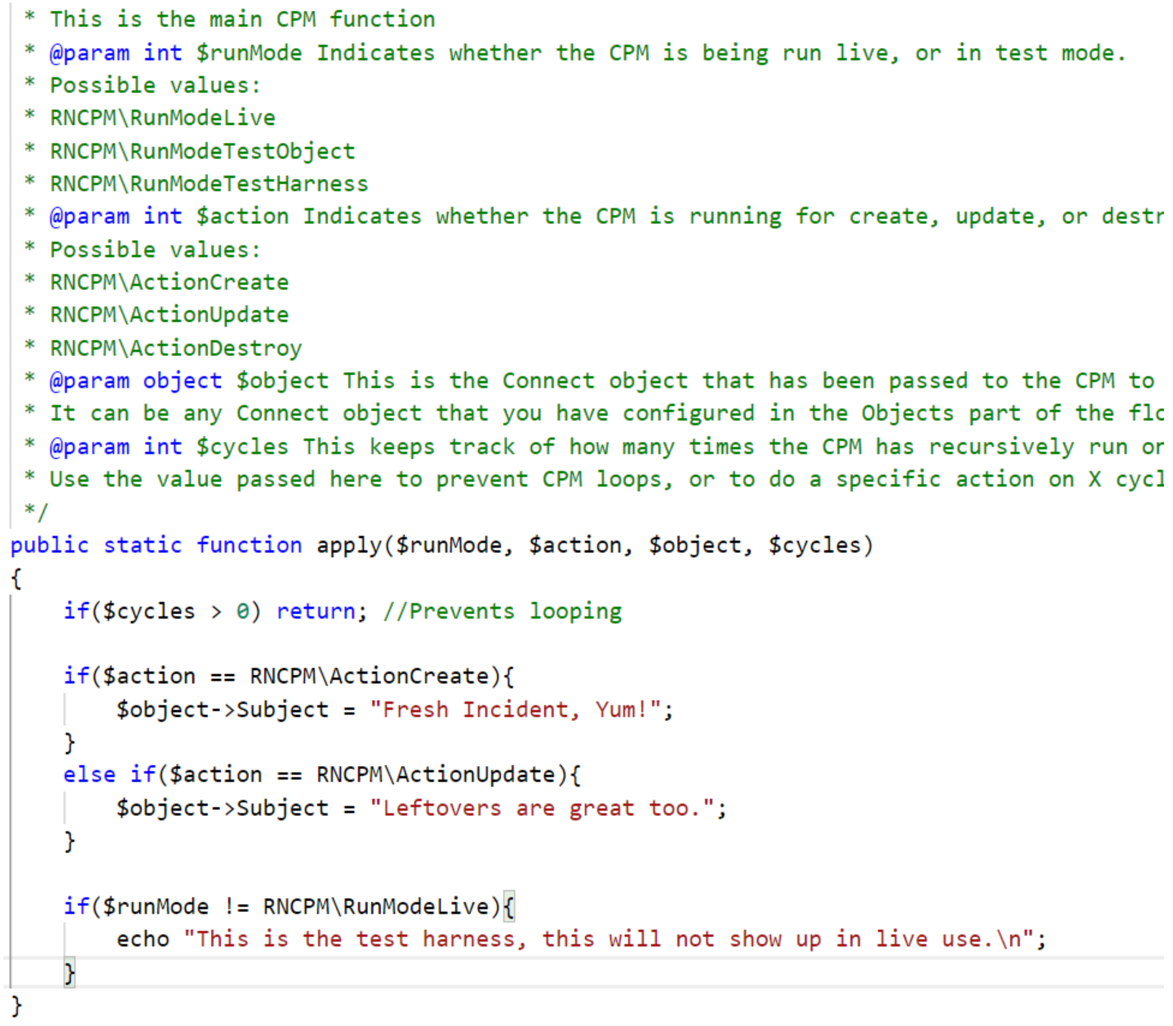
```php
/**
 * This function runs after setup(). It will run once for each type of object and each type
 * of action as configured in the Objects and Actions sections of the flower basket.
 * Example: If you have two Objects and two Actions configured, this function will run four times.
 * @param int $action Indicates whether a create, update, or destroy action is being tested.
 * Possible values:
 * RNCPM\ActionCreate
 * RNCPM\ActionUpdate
 * RNCPM\ActionDestroy
 * @param class $objectType Indicates which object type is being tested.
 * @return mixed Return a Connect object of the expected type. This object will
 * be processed through the apply() function.
 * You can also pass an array of Connect objects, in which case each item in the array
 * will separately pass through the apply() function.
 */
public static function fetchObject($action, $objectType)
{
    if($objectType == "RightNow\Connect\v1_3\Incident"){
        return static::$testIncident;
    }
    else if($objectType == "RightNow\Connect\v1_3\Contact"){
        return static::$testContact;
    }
}
```

Function: apply()



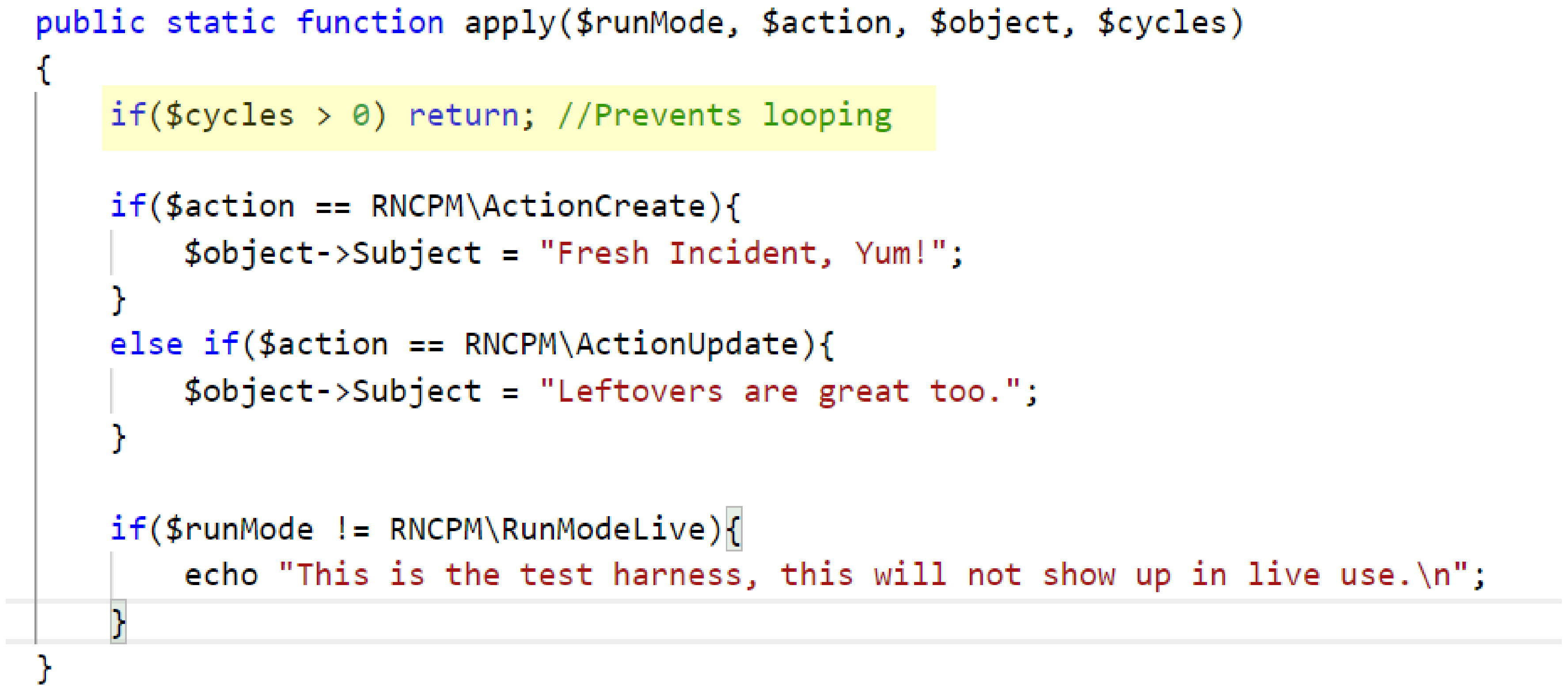
```
<?
/**
 * CPMObjectEventHandler: CPMExample
 * Package: RN
 * Objects: Incident, Contact
 * Actions: Create, Update
 * Version: 1.3
 */

use \RightNow\CPM\v1 as RNCPM,
\RightNow\Connect\v1_3 as RNCPHP; //Connect version specified here must correspond with the version specified in Version in the flower basket

class CPMExample implements RNCPM\ObjectEventHandler
{


    public static function apply($runMode, $action, $object, $cycles)
    {

    }
}

class CPMExample_TestHarness implements RNCPM\ObjectEventHandler_TestHarness
{

    public static function setup()
    {

    }

    public static function fetchObject($action, $objectType)
    {

    }

    public static function validate($action, $object)
    {

    }

    public static function cleanup()
    {

    }
}
```

Function: apply()



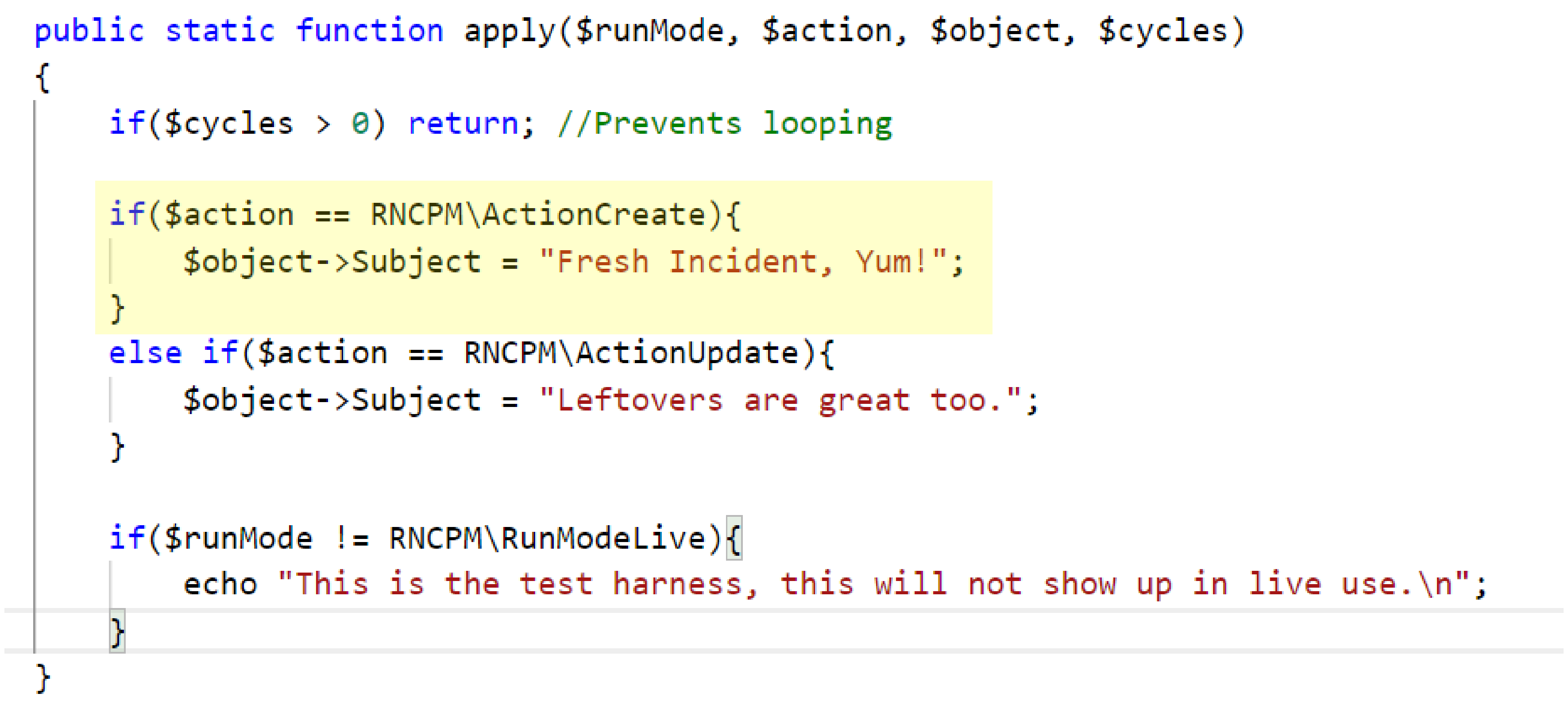
```
 * This is the main CPM function
 * @param int $runMode Indicates whether the CPM is being run live, or in test mode.
 * Possible values:
 * RNCPM\RunModeLive
 * RNCPM\RunModeTestObject
 * RNCPM\RunModeTestHarness
 * @param int $action Indicates whether the CPM is running for create, update, or destr
 * Possible values:
 * RNCPM\ActionCreate
 * RNCPM\ActionUpdate
 * RNCPM\ActionDestroy
 * @param object $object This is the Connect object that has been passed to the CPM to
 * It can be any Connect object that you have configured in the Objects part of the flc
 * @param int $cycles This keeps track of how many times the CPM has recursively run or
 * Use the value passed here to prevent CPM loops, or to do a specific action on X cycl
 */
public static function apply($runMode, $action, $object, $cycles)
{
    if($cycles > 0) return; //Prevents looping

    if($action == RNCPM\ActionCreate){
        $object->Subject = "Fresh Incident, Yum!";
    }
    else if($action == RNCPM\ActionUpdate){
        $object->Subject = "Leftovers are great too.";
    }

    if($runMode != RNCPM\RunModeLive){
        echo "This is the test harness, this will not show up in live use.\n";
    }
}
```

Function: apply()



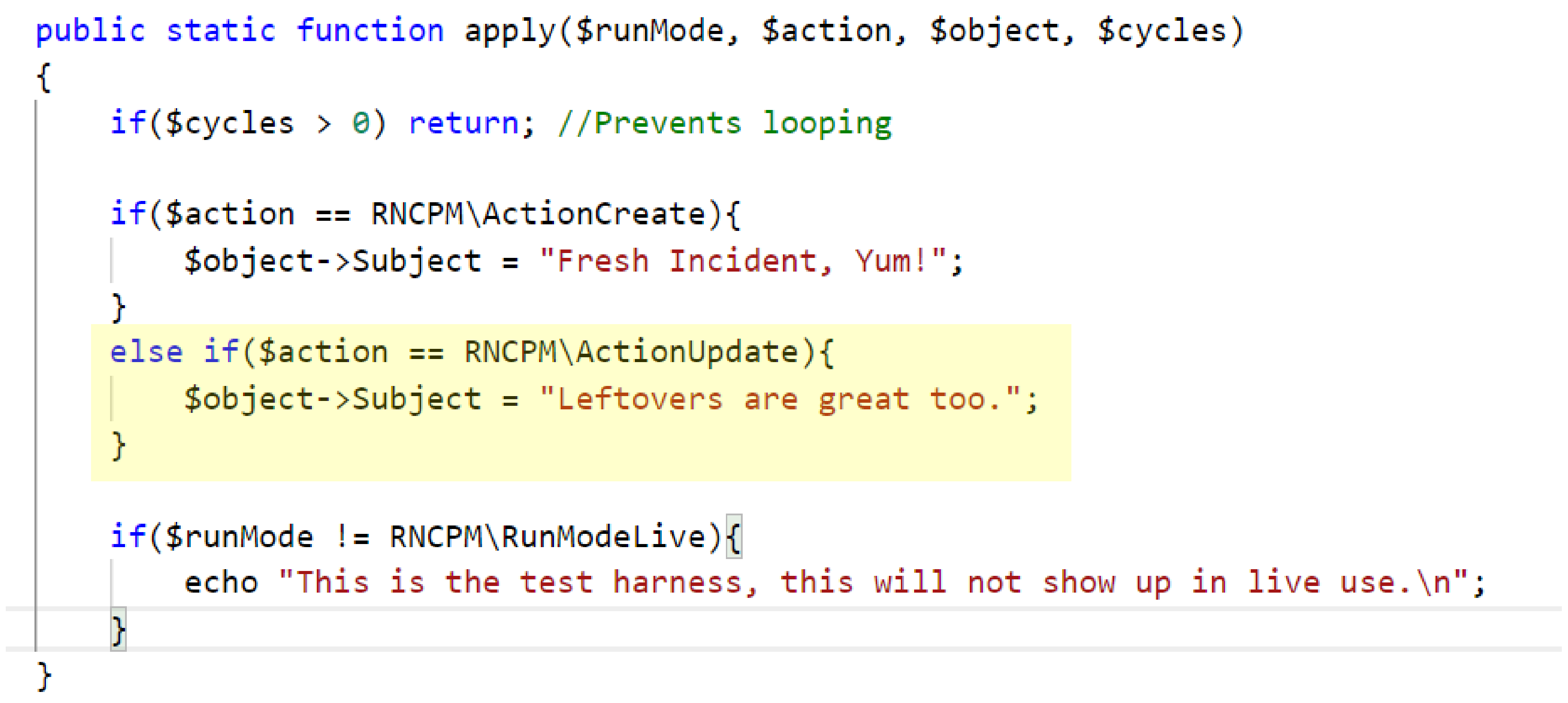
```
public static function apply($runMode, $action, $object, $cycles)
{
    if($cycles > 0) return; //Prevents looping

    if($action == RNCPM\ActionCreate){
        $object->Subject = "Fresh Incident, Yum!";
    }
    else if($action == RNCPM\ActionUpdate){
        $object->Subject = "Leftovers are great too.";
    }

    if($runMode != RNCPM\RunModeLive){
        echo "This is the test harness, this will not show up in live use.\n";
    }
}
```

Function: apply()



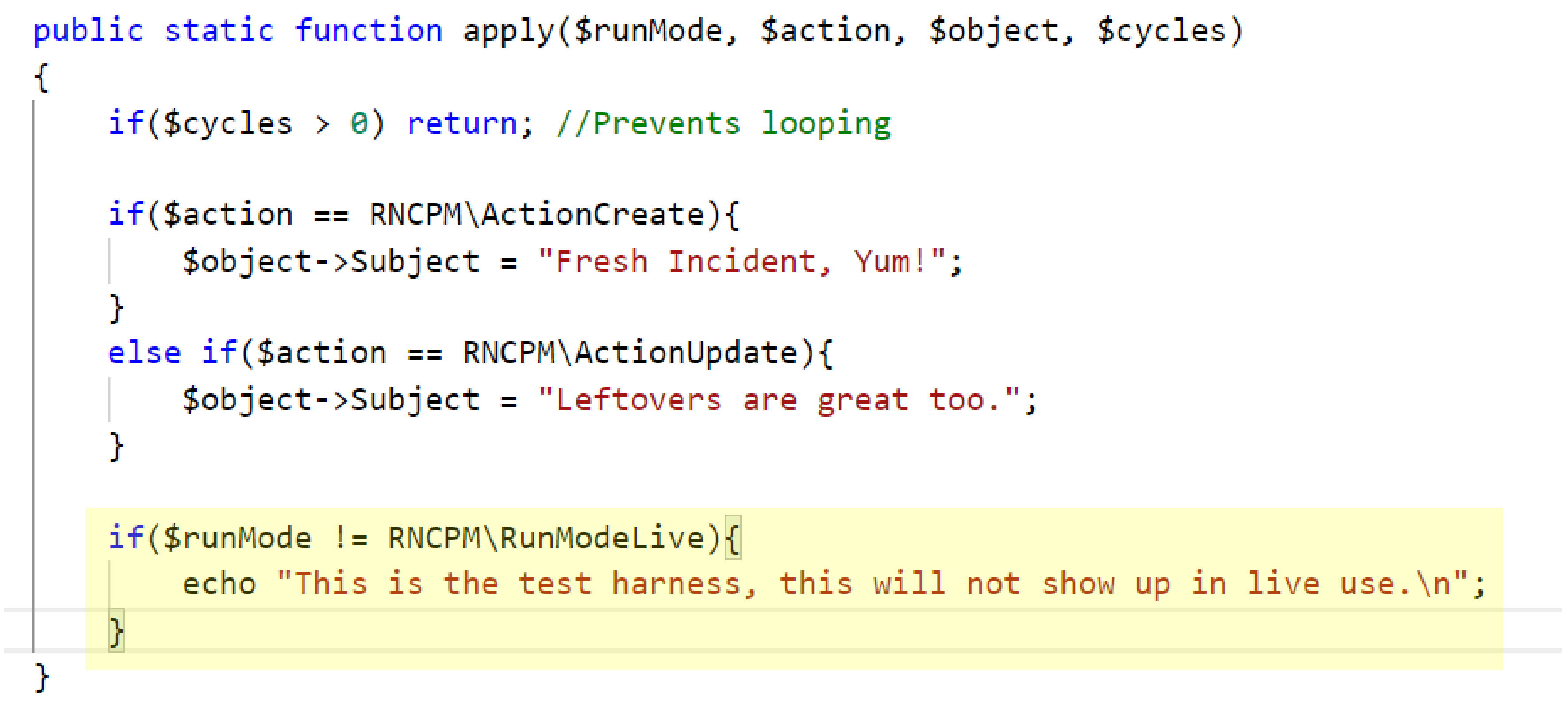
```php
public static function apply($runMode, $action, $object, $cycles)
{
    if($cycles > 0) return; //Prevents looping

    if($action == RNCPM\ActionCreate){
        $object->Subject = "Fresh Incident, Yum!";
    }
    else if($action == RNCPM\ActionUpdate){
        $object->Subject = "Leftovers are great too.";
    }

    if($runMode != RNCPM\RunModeLive){
        echo "This is the test harness, this will not show up in live use.\n";
    }
}
```

Function: apply()



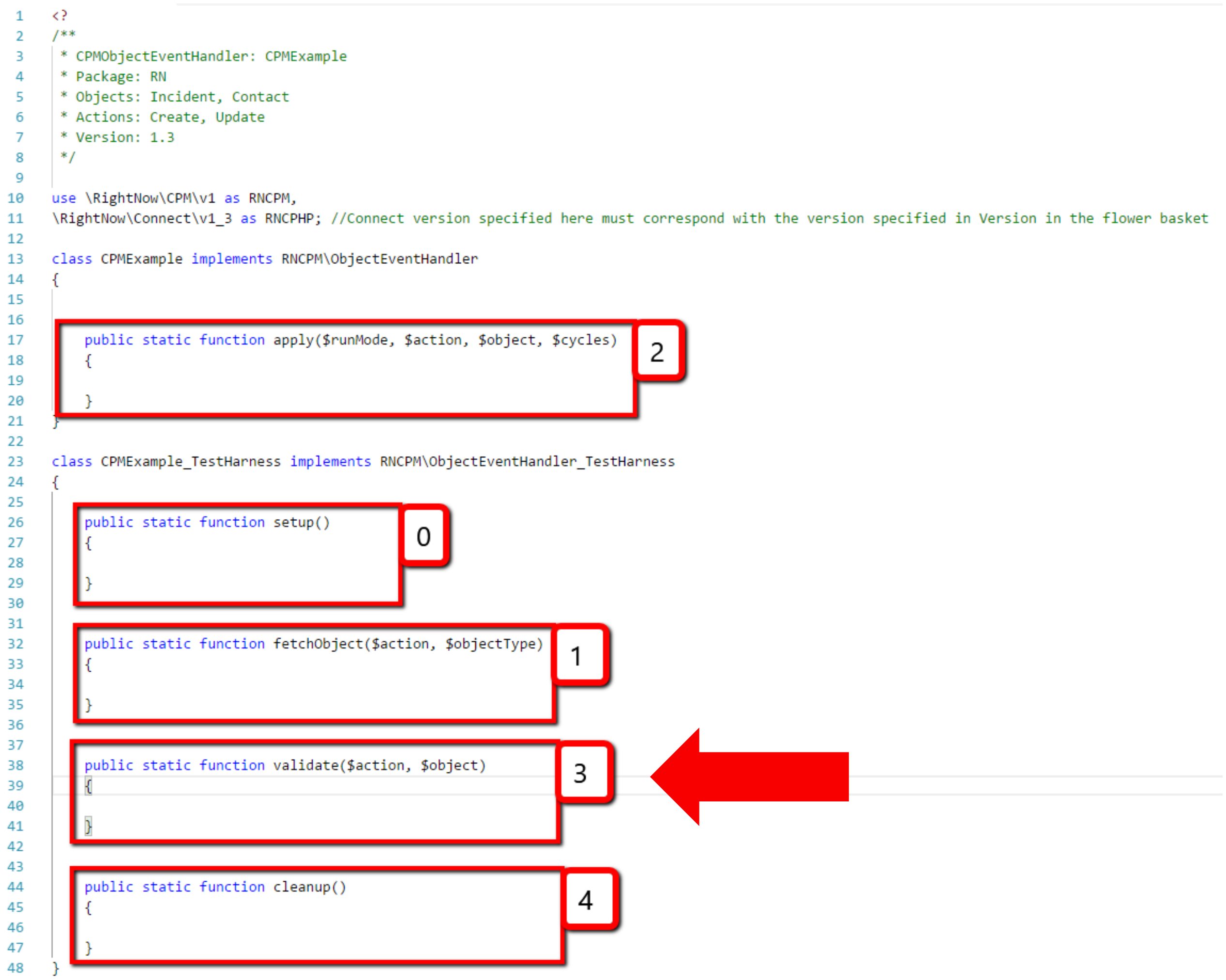
```
public static function apply($runMode, $action, $object, $cycles)
{
    if($cycles > 0) return; //Prevents looping

    if($action == RNCPM\ActionCreate){
        $object->Subject = "Fresh Incident, Yum!";
    }
    else if($action == RNCPM\ActionUpdate){
        $object->Subject = "Leftovers are great too.";
    }

    if($runMode != RNCPM\RunModeLive){
        echo "This is the test harness, this will not show up in live use.\n";
    }
}
```

Function: apply()



```
public static function apply($runMode, $action, $object, $cycles)
{
    if($cycles > 0) return; //Prevents looping

    if($action == RNCPM\ActionCreate){
        $object->Subject = "Fresh Incident, Yum!";
    }
    else if($action == RNCPM\ActionUpdate){
        $object->Subject = "Leftovers are great too.";
    }

    if($runMode != RNCPM\RunModeLive){
        echo "This is the test harness, this will not show up in live use.\n";
    }
}
```

Function: validate()



```php
<?
/**
 * CPMObjectEventHandler: CPMExample
 * Package: RN
 * Objects: Incident, Contact
 * Actions: Create, Update
 * Version: 1.3
 */

use \RightNow\CPM\v1 as RNCPM,
\RightNow\Connect\v1_3 as RNCPHP; //Connect version specified here must correspond with the version specified in Version in the flower basket

class CPMExample implements RNCPM\ObjectEventHandler
{


    public static function apply($runMode, $action, $object, $cycles)
    {

    }
}

class CPMExample_TestHarness implements RNCPM\ObjectEventHandler_TestHarness
{

    public static function setup()
    {

    }

    public static function fetchObject($action, $objectType)
    {

    }

    public static function validate($action, $object)
    {

    }

    public static function cleanup()
    {

    }
}
```

Function: validate()

```php
public static function validate($action, $object)
{
    $success = true;
    if($action == RNCPM\ActionCreate){
        $success = ($object->Subject == "Fresh Incident, Yum!");
        if($success){
            echo "Passed Create Test\n";
        }
        else {
            echo "Failed Create Test\n Subject was: {$object->Subject}\n";
        }
    }

    else if($action == RNCPM\ActionUpdate){
        $success = (strpos($object->Subject, "Leftovers are great too.") !== false);
        if($success){
            echo "Passed Update Test.\n";
        }
        else {
            echo "Failed Update Test\n Subject was: {$object->Subject}\n";
        }
    }

    return $success;
}
```

```php
public static function validate($action, $object)
{
    $success = true;
    if($action == RNCPM\ActionCreate){
        $success = ($object->Subject == "Fresh Incident, Yum!");
        if($success){
            echo "Passed Create Test\n";
        }
        else {
            echo "Failed Create Test\n Subject was: {$object->Subject}\n";
        }
    }

    else if($action == RNCPM\ActionUpdate){
        $success = (strpos($object->Subject, "Leftovers are great too.") !== false);
        if($success){
            echo "Passed Update Test.\n";
        }
        else {
            echo "Failed Update Test\n Subject was: {$object->Subject}\n";
        }
    }

    return $success;
}
```

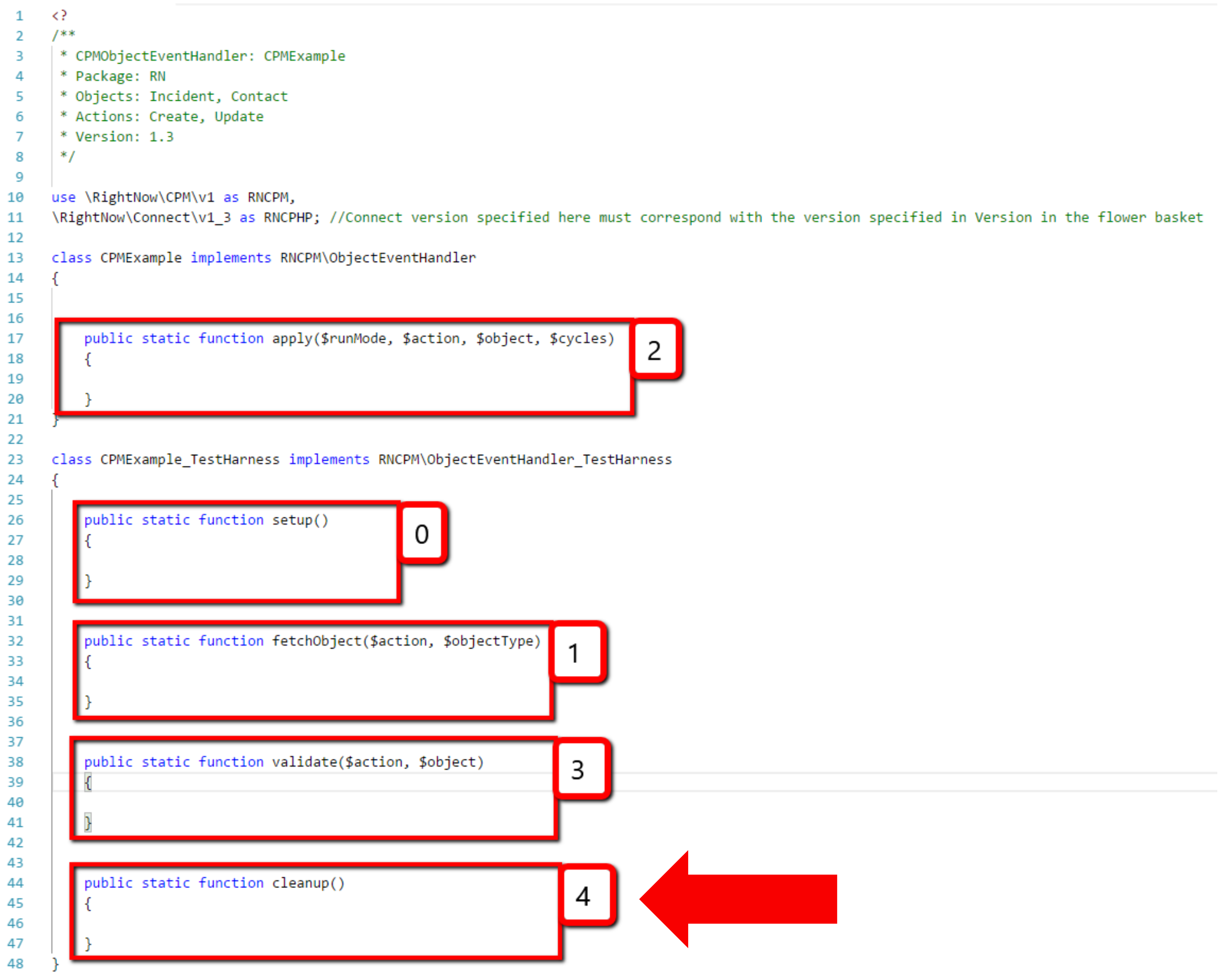
```php
public static function validate($action, $object)
{
    $success = true;
    if($action == RNCPM\ActionCreate){
        $success = ($object->Subject == "Fresh Incident, Yum!");
        if($success){
            echo "Passed Create Test\n";
        }
        else {
            echo "Failed Create Test\n Subject was: {$object->Subject}\n";
        }
    }

    else if($action == RNCPM\ActionUpdate){
        $success = (strpos($object->Subject, "Leftovers are great too.") !== false);
        if($success){
            echo "Passed Update Test.\n";
        }
        else {
            echo "Failed Update Test\n Subject was: {$object->Subject}\n";
        }
    }

    return $success;
}
```

ORACLE

```php
public static function validate($action, $object)
{
    $success = true;
    if($action == RNCPM\ActionCreate){
        $success = ($object->Subject == "Fresh Incident, Yum!");
        if($success){
            echo "Passed Create Test\n";
        }
        else {
            echo "Failed Create Test\n Subject was: {$object->Subject}\n";
        }
    }

    else if($action == RNCPM\ActionUpdate){
        $success = (strpos($object->Subject, "Leftovers are great too.") !== false);
        if($success){
            echo "Passed Update Test.\n";
        }
        else {
            echo "Failed Update Test\n Subject was: {$object->Subject}\n";
        }
    }

    return $success;
}
```

```php
public static function validate($action, $object)
{
    $success = true;
    if($action == RNCPM\ActionCreate){
        $success = ($object->Subject == "Fresh Incident, Yum!");
        if($success){
            echo "Passed Create Test\n";
        }
        else {
            echo "Failed Create Test\n Subject was: {$object->Subject}\n";
        }
    }

    else if($action == RNCPM\ActionUpdate){
        $success = (strpos($object->Subject, "Leftovers are great too.") !== false);
        if($success){
            echo "Passed Update Test.\n";
        }
        else {
            echo "Failed Update Test\n Subject was: {$object->Subject}\n";
        }
    }

    return $success;
}
```

Function: cleanup()

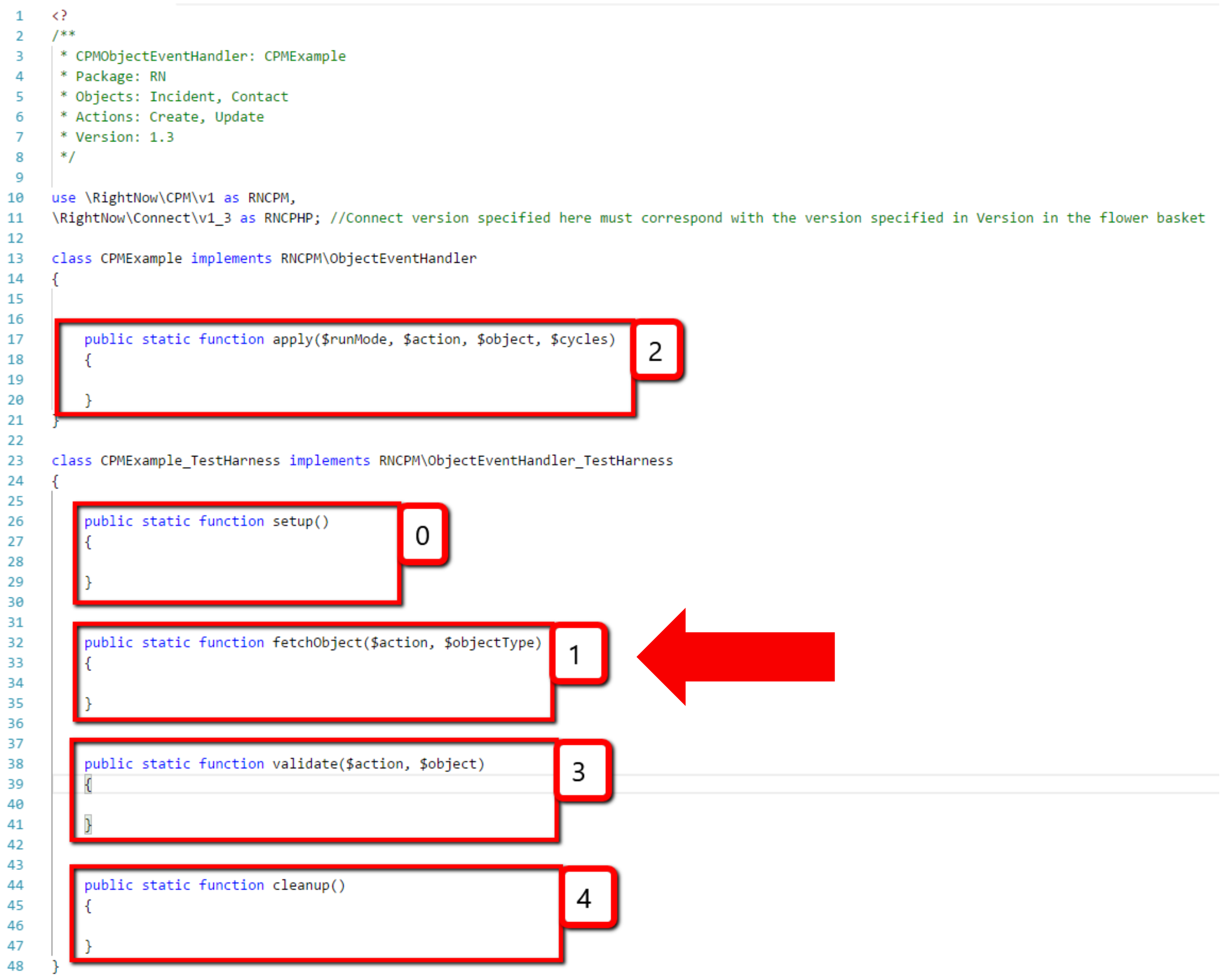


```
<?
/**
 * CPMObjectEventHandler: CPMExample
 * Package: RN
 * Objects: Incident, Contact
 * Actions: Create, Update
 * Version: 1.3
 */

use \RightNow\CPM\v1 as RNCPM,
\RightNow\Connect\v1_3 as RNCPHP; //Connect version specified here must correspond with the version specified in Version in the flower basket

class CPMExample implements RNCPM\ObjectEventHandler
{


    public static function apply($runMode, $action, $object, $cycles)
    {

    }
}

class CPMExample_TestHarness implements RNCPM\ObjectEventHandler_TestHarness
{

    public static function setup()
    {

    }

    public static function fetchObject($action, $objectType)
    {

    }

    public static function validate($action, $object)
    {

    }

    public static function cleanup()
    {

    }
}
```

Function: cleanup()

```
/**
 * This function runs once per validate() run.
 * Seldom used, except in cases where external data may have
 * been modified during a test run (via external API calls, etc.)
 */
public static function cleanup()
{

}
```

Back to fetchObject



```
<?
/**
 * CPMObjectEventHandler: CPMExample
 * Package: RN
 * Objects: Incident, Contact
 * Actions: Create, Update
 * Version: 1.3
 */

use \RightNow\CPM\v1 as RNCPM,
\RightNow\Connect\v1_3 as RNCPHP; //Connect version specified here must correspond with the version specified in Version in the flower basket

class CPMExample implements RNCPM\ObjectEventHandler
{


    public static function apply($runMode, $action, $object, $cycles)
    {

    }
}

class CPMExample_TestHarness implements RNCPM\ObjectEventHandler_TestHarness
{

    public static function setup()
    {

    }

    public static function fetchObject($action, $objectType)
    {

    }

    public static function validate($action, $object)
    {

    }

    public static function cleanup()
    {

    }
}
```

Debugging

Development and Live Usage

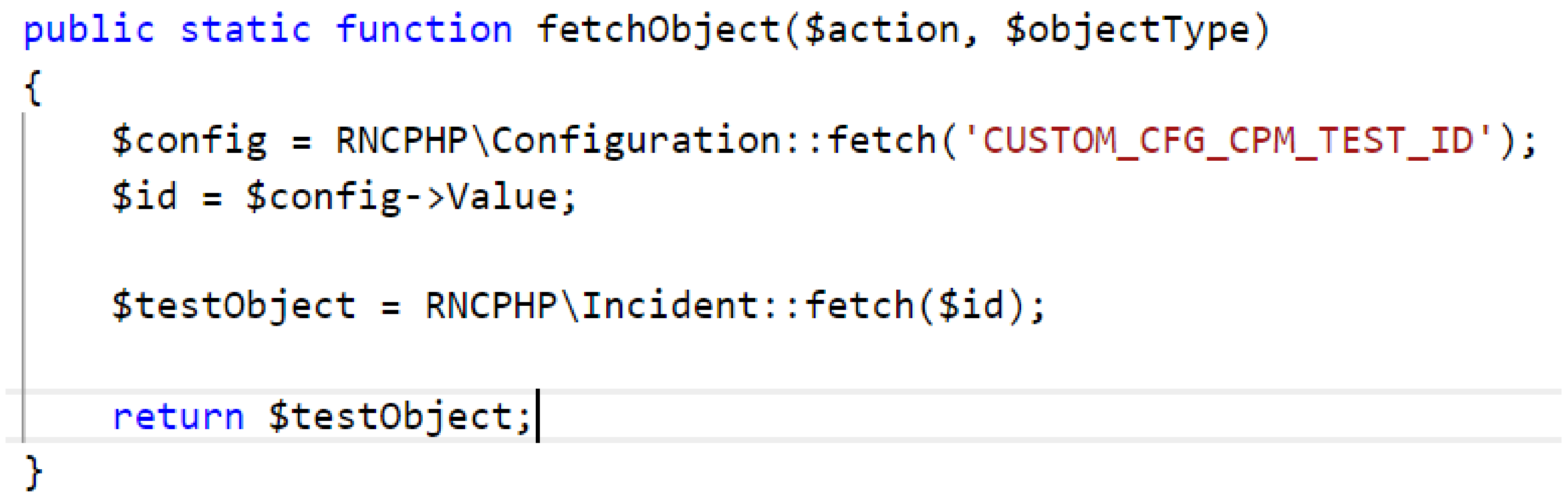
Debugging During Development

- Use echo, var_dump, etc.
  - Use $runMode logic to enable verbose debugging
- In case of silent failure, check PHP syntax
- Check error and info logs
- Remember that changes are not committed during testing
- Public Mail API will send
- PHP debugging functions – debug_backtrace(), etc.

Debugging During Live Use

- Catch exceptions and failures
  - Log to Custom Object
  - Send error notifications with Public Mail API
    - Attach output from debug_backtrace(), see later example
  - Log ID of object that produced failure
- Use a custom config to store a testing ID
  - Allows easy testing when a given object fails to process

Example: Config-Stored Testing ID



```
public static function fetchObject($action, $objectType)
{
    $config = RNCPHP\Configuration::fetch('CUSTOM_CFG_CPM_TEST_ID');
    $id = $config->Value;

    $testObject = RNCPHP\Incident::fetch($id);

    return $testObject;
}
```

Example: Sending Error Data w/Mail API

```php
public static function sendErrorData(){
    $mm = new RNCPHP\MailMessage();
    $mm->To->EmailAddresses = array("some.one@your_site.com");
    $mm->Subject = "CPM Error Report";

    $exceptionData = json_encode(debug_backtrace());
    $fattach = new RNCPHP\FileAttachment();
    $fattach->ContentType = "text/text";
    $fp = $fattach->makeFile();
    fwrite($fp, $exceptionData);
    fclose($fp);
    $fattach->FileName = "ExceptionData.txt";
    $fattach->Name = "ExceptionData.txt";

    $mm->FileAttachments = new RNCPHP\FileAttachmentArray();
    $mm->FileAttachments[] = $fattach;

    $mm->Body->Text = "Exception Data Attached";
    $mm->Body->Html = "Exception Data Attached";

    $mm->send();
}
```

Advanced CPMs

Asynchronous CPMs

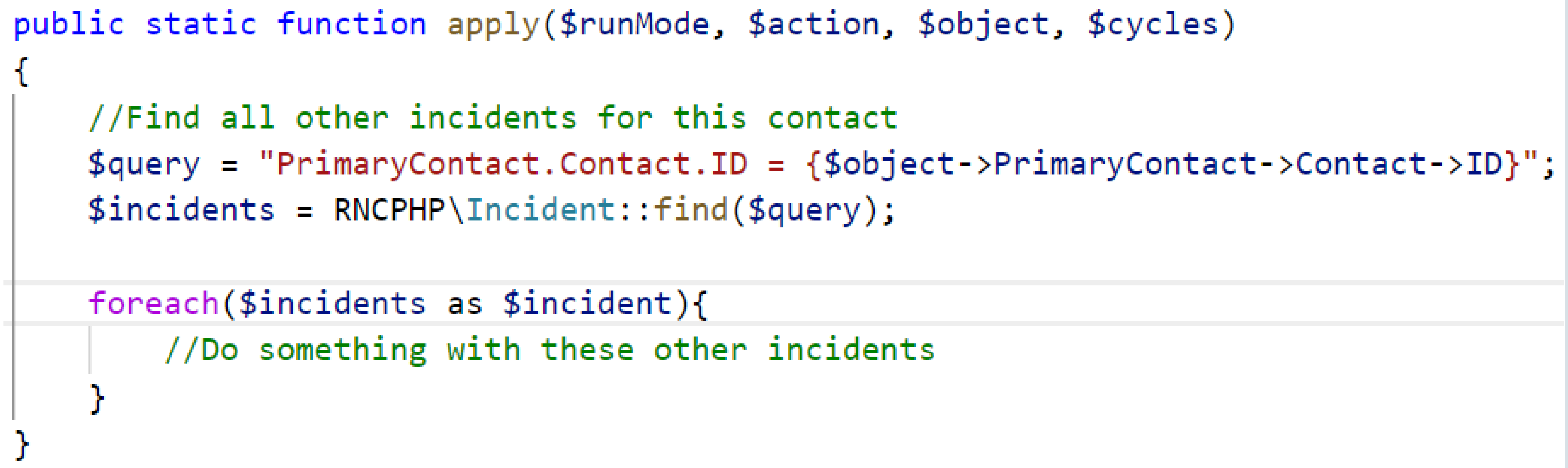
- Automatically retries failed runs
  - Retries occur at 90, 450, 3,150, and 34,650 seconds after initial attempt
- Non-blocking, will not prevent save operations from completing while it runs
- Required for CPMs that utilize cURL
- Recommended for long-running scripts

Using the “prev” Object w/Test Harness

Example:

```php
public static function fetchObject($action, $objectType)
{
    if($action == RNCPM\ActionUpdate){
        $prevIncident = new RNCPHP\Incident();
        $prevIncident->PrimaryContact = static::$testContact;
        //Set whatever prev fields we care about
        $prevIncident->StatusWithType->Status->ID = 3;

        static::$testIncident->prev = $prevIncident; //Now our test incident has a prev object
    }

    return static::$testIncident;
}
```

ROQL / Object Retrieval Testing



```php
public static function apply($runMode, $action, $object, $cycles)
{
    //Find all other incidents for this contact
    $query = "PrimaryContact.Contact.ID = {$object->PrimaryContact->Contact->ID}";
    $incidents = RNCPHP\Incident::find($query);

    foreach($incidents as $incident){
        //Do something with these other incidents
    }
}
```

ROQL / Object Retrieval Testing

```
public static function setup()
{
    $contact = new RNCPHP\Contact();
    $contact->Name = new RNCPHP\PersonName();
    $contact->Name->First = "Malcolm";
    $contact->Name->Last = "Reynolds";
    $contact->save();
    static::$testContact = $contact;

    //Create additional incident(s) to query for:
    $otherContactIncident = new RNCPHP\Incident();
    $otherContactIncident->PrimaryContact = $contact;
    $otherContactIncident->save();

    //Commit incident(s)
    RNCPHP\ConnectAPI::commit();

    $incident = new RNCPHP\Incident();
    $incident->PrimaryContact = $contact;
    $incident->Subject = "Test Incident";

    static::$testIncident = $incident;
}
```

Q&A

Safe Harbor Statement