# MODULE 4: SOFWARE CODING

---

# 1. Software Coding Standards in PHP

### What Are Coding Standards?

Coding standards are a set of **rules and guidelines** that developers follow to write **clean**, **readable**, **secure**, and **maintainable** code.

---

### PHP Coding Standards Examples:

## PHP Standard Recommendation

#### a) PSR-1: Basic Coding Standard

- File must start with `<?php`
- Class names must be in **PascalCase**: `UserProfile`
- Method names in **camelCase**: `getUserDetails()`
- Constants in **UPPERCASE**: `MAX_ATTEMPTS`

#### b) PSR-12: Extended Coding Style

- **Indentation**: Use 4 spaces (no tabs)
- **Line length**: Max 120 characters
- **No closing `?>` tag** in pure PHP files
- Proper **namespace** and **import** usage

---

### Example – Bad Code:

```
class user{public function Getname($N){return $N;}}
```

### Example – Good Code (PSR-12 Compliant):

```
<?php

namespace App\Models;

class User
{
    public function getName(string $name): string
    {
        return $name;
    }
}
```

---

### Helpful Tools:

- **PHP_CodeSniffer**: Automatically checks your code for PSR violations.

```
composer require --dev squizlabs/php_codesniffer
./vendor/bin/phpcs --standard=PSR12 yourfile.php
```

---

# 2. Testing and Debugging in PHP

## A. Software Testing

### Purpose:

Testing ensures that **code behaves as expected**, and helps catch bugs early.

---

### Types of Tests in PHP:

| Type | Description | Tool Example |
|---|---|---|
| **Unit Testing** | Test individual functions/classes | PHPUnit |
| **Integration** | Test multiple components together | Codeception |
| **Functional** | Test user interface and flow | Behat, Selenium |

---

### Example – PHPUnit Test (Unit Testing):

1. Install PHPUnit:

```
composer require --dev phpunit/phpunit
```

2. Write Function to Test:

```
function multiply(int $a, int $b): int
{
    return $a * $b;
}
```

3. Write Test Case:

```
use PHPUnit\Framework\TestCase;

class MathTest extends TestCase
{
    public function testMultiply()
    {
        $this->assertEquals(20, multiply(4, 5));
    }
}
```

4. Run Test:

```
./vendor/bin/phpunit tests/MathTest.php
```

---

# B. Debugging in PHP

## Common Debugging Techniques:

| Technique | Example |
|---|---|
| `var_dump()` | `var_dump($data);` – Shows type and value of variable |
| `print_r()` | Prints arrays/objects |
| `error_log()` | Logs message to server log: `error_log("Issue here");` |
| **Xdebug** | Advanced debugger: breakpoints, step-through, profiling |
| **Browser Dev Tools** | Inspect network calls, console logs (for frontend issues) |

---

## Example Debugging Code:

```
$user = ['name' => 'John', 'email' => 'john@example.com'];
var_dump($user);
```

## Enabling Error Reporting:

```
ini_set('display_errors', 1);
error_reporting(E_ALL);
```

---

## Xdebug Example:

- Setup with VSCode or PhpStorm
- Add breakpoints and watch variable values live

---

# 3. Code Review in PHP

## What is Code Review?

Code review is a **peer evaluation of code** to ensure it meets standards, is free from bugs, and is efficient.

---

### Code Review Checklist:

| Criteria | Example/Explanation |
|---|---|
| **Readability** | Is code easy to understand? |
| **Security** | Is user input sanitized? SQL injection protected? |
| **Efficiency** | Is the logic optimized? Any redundant code? |
| **Compliance** | Does code follow PSR-12 or project standards? |
| **Test Coverage** | Are there sufficient tests for new code? |

---

### Example Feedback:

- *Bad:* `mysqli_query($conn, "SELECT * FROM users WHERE id=$id");`
- *Feedback:* Use prepared statements to prevent SQL injection.

---

### Tools for Code Review:

- **GitHub/GitLab/Bitbucket** – Pull requests and inline comments.
- **PHP_CodeSniffer** – Automated coding standard checks.
- **SonarQube** – Advanced static code analysis.

---

### Example Git Workflow:

1. Developer pushes code to `feature-branch`.
2. Creates Pull Request to `main`.
3. CI/CD runs tests and PHP_CodeSniffer.
4. Reviewer suggests changes.
5. Code merged after approval.

---

# 4. CASE Tools in PHP Development

## What Are CASE Tools?

CASE (Computer-Aided Software Engineering) tools **automate and support software development** tasks such as coding, testing, documentation, deployment, and project management.

---

### Categories and Examples:

| Category | Tool Example | Purpose |
|---|---|---|
| **IDE** | PhpStorm, VS Code | Code editing, syntax highlighting, debugging |
| **Version Control** | Git, GitHub, GitLab | Track changes, collaborate, pull requests |
| **Testing** | PHPUnit, Codeception | Unit and integration testing |
| **Debugging** | Xdebug, PHP Debug Bar | Trace code, inspect variables, profiling |
| **Documentation** | phpDocumentor | Generate API documentation from code |
| **CI/CD** | Jenkins, GitHub Actions | Automate testing, building, and deployment |
| **Project Mgmt** | Jira, Trello | Manage tasks, track issues |

---

### Example CASE Workflow:

1. **PhpStorm IDE** – Write code.
2. **GitHub** – Push code; pull request triggers CI.
3. **GitHub Actions** – Run PHPUnit, PHP_CodeSniffer.
4. **Xdebug** – Debug failed test locally.
5. **phpDocumentor** – Generate docs for API endpoints.

---

### phpDocumentor Example:

```
composer require --dev phpdocumentor/phpdocumentor
phpdoc run -d src -t docs
```

---

# Summary Table

| Concept | Tools/Examples |
|---|---|
| **Coding Standards** | PSR-1, PSR-12, PHP_CodeSniffer |
| **Testing** | PHPUnit (unit tests), Codeception, Behat |
| **Debugging** | var_dump(), error_log(), Xdebug |
| **Code Review** | GitHub PRs, Bitbucket, GitLab CI/CD |
| **CASE Tools** | PhpStorm, Git, Jenkins, phpDocumentor, Jira |

# Final Thoughts:

- **Consistency** in coding (via standards) saves time and prevents errors.
- **Testing and debugging** ensure reliability and confidence in your software.
- **Code reviews** enhance collaboration, security, and maintainability.
- **CASE tools** automate tasks, streamline workflows, and boost productivity.