Web Application for Personalised Learning of Object Oriented Programming

Hock Chue Ha Alicia

Ngee Ann Polytechnic, School of Engineering, Singapore

hch5@np.edu.sg

Object Oriented Programming is a module taken by all Year 2 students from Electronic & Computer Engineering, Aerospace Engineering and Engineering Science Diploma courses, in Ngee Ann Polytechnic (NP). This module builds on the programming fundamentals taught in the Year 1 Programming module.

Object Oriented Programming is difficult to learn as students must shift their thinking to conceptualize real-world objects in code, understand relationship between objects and master complex concepts like classes, encapsulation, inheritance, and polymorphism.

Student programming competencies within a class can vary significantly. When uniform material and exercises are assigned, less experienced students struggle and require additional support, while advanced learners find the content insufficiently challenging. This one-size-fits-all approach often leads to learning dissatisfaction across different competency levels.

A web application, CSHARPY, is developed to teach Object Oriented Programming in C#. CSHARPY offers a personalized learning approach which allows students to choose how to learn. CSHARPY features two distinct learning modes. The first mode provides line-by-line guidance, supporting less experienced students to complete the exercises successfully while building their motivation to continue learning. The second mode is designed for students with stronger programming foundations, who can complete the exercises on their own. This advanced mode also analyses students' entire programs and provides feedback on their mistakes, allowing proficient students to complete more challenging exercises and further develop their skills.

A survey in BrightSpace (NP's Learning Management System) is conducted and the results show that most students found CSHARPY helpful. This module adopts a Constructivist learning pedagogy, where students build their own understanding through hands-on programming experience. Instead of lectures, each lesson engages students in active learning through writing code, testing and debugging the programs. The module follows an incremental learning progression where students write a first program to create a class, followed by a second program to create a class and property, and a third program to create a class, property and method. Each program illustrates real-world application of Object Oriented Programming, encouraging students to connect new concepts with their existing knowledge and incrementally build their understanding.

Keywords: *Programming, Constructivism, Personalised Learning*

Introduction

Object Oriented Programming can be challenging to learn and understand, primarily due to its abstract nature, the shift in mindset from procedural programming, and the need to conceptualize real-world entities as code (Kolling, 1999). Key concepts such as inheritance, polymorphism, and encapsulation often intimidate beginners, and applying design patterns effectively is a skill that takes both time and experience to develop. Object Oriented Programming demands a higher level of cognitive ability, particularly in abstract reasoning and complex problem-solving. A solid foundation in core programming principles, algorithms, and data structures is crucial for understanding and mastering Object Oriented Programming concepts.

At Ngee Ann Polytechnic, Year 2 students from Diploma of Electronic & Computer Engineering, exhibit a broad range of strengths and skill levels, with many lacking a strong understanding of fundamental programming principles, possibly due to lack of practice. These students may struggle significantly when learning Object Oriented Programming, requiring close guidance and step-by-step assistance in writing even basic lines of code. Given that Object Oriented Programming involves learning new syntax and tackling lengthy, intricate programs, it can be especially difficult

for instructors to provide effective support in traditional classroom settings.

On the other hand, some students demonstrate advanced programming capabilities and can independently complete exercises. These students benefit from immediate feedback on their work, allowing them to refine their skills efficiently. To maintain their engagement and stimulate critical thinking, these students require more challenging problems and advanced exercises.

Given the diverse proficiency levels among students, a one-size-fits-all teaching approach is ineffective. Differentiated instruction methods can better cater to individual students.Personalized Learning, which has shown promising results in programming education (Junus, 2017), offers a potential solution by enabling students to choose and engage with the material according to their individual needs. Recognizing that each student has unique learning preferences and abilities, research indicates that adapting teaching approaches to accommodate different learning preferences and abilities can improve student outcomes. Hence, wo distinct learning paths were developed while covering the same core concepts, allowing students to progress at their own pace.

CSHARPY: Design and Implementation

CSHARPY is a web application designed to teach Object Oriented Programming based on C#. The CSharpy web app is published to cloud services. It communicates with the Google Gemini AI model using the Google Gen AI SDK. The system implements a personalized learning approach grounded in constructivist educational theory. The CSHARPY web application presents users with a landing page (Figure 1) that offers two distinct learning modes to cater to students with varying levels of programming proficiency and confidence. This initial interface allows students to choose between independent practice with immediate feedback (Mode 1) or step-by-step guided learning (Mode 2).

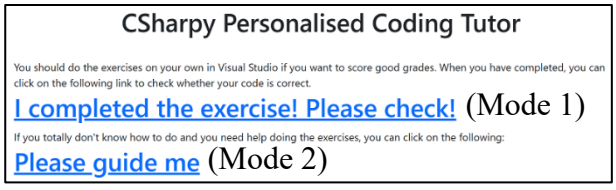

Figure 1: CSHARPY Web Application's homepage showing the two learning mode options

Mode 1: Independent Practice with Immediate Feedback

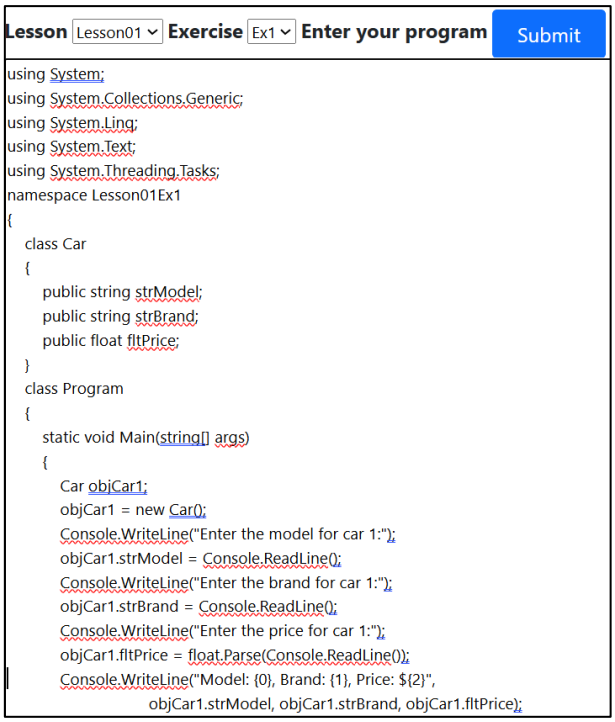

Figure 2 CSHARPY Mode 1

This mode supports students who can complete coding exercises independently and would like immediate feedback from CSHARPY. In this mode:

- Students paste their completed code into the provided text area in CSHARPY (Figure 2) and click **Submit**.
- CSHARPY utilizes the Google Gemini AI model to analyse the submitted code, performing both syntactic and logical error detection.
- When errors are detected, the system provides **detailed, targeted feedback** that explains the issue and suggests specific corrections.

Example 1: Logical Error – Incorrect Object Reference

```
Car objCar2 = new Car();
Console.WriteLine("Enter the model for car 2:");
objCar1.strModel = Console.ReadLine();
// Incorrect object reference
```

CSHARPY Response:

The program contains the following mistake:
On line 40:
objCar1.strModel = Console.ReadLine();

should be

objCar2.strModel = Console.ReadLine();.

Example 2: Constructor Argument Order

```
public Computer(string brand, string shopname) : base(shopname)
    {
       strBrand = brand;
    }

objCom = new Computer(shopname, brand);
// Incorrect argument order class
```

CSHARPY Response:

The student program has one mistake:
In the `Program` class, inside the `Main` method, the `Computer` object is instantiated incorrectly. The arguments passed to the `Computer` constructor are in the wrong order:

```
objCom = new Computer(shopname, brand);
```

It should be:

```
objCom = new Computer(brand, shopname);
```

Example 3: Flexibility in Coding Style

CSHARPY recognizes and accepts semantically equivalent code variations for tasks like object declaration and instantiation.

C#

```
// Two-line approach

Car objCar1;

objCar1 = new Car();

// One-line approach

Car objCar1 = new Car();
```

CSHARPY accepts that both single-line and two-line approaches are semantically equivalent in C#. Both approaches achieve the same outcome and the system refrain from flagging either one as an error.

The system's immediate feedback mechanism aligns with established principles of constructivist learning. By enabling rapid error identification and correction, it allows students to progressively tackle more challenging exercises and deepening their understanding of Object Oriented Programming concepts.

Mode 2: Step-by-step Guided Learning

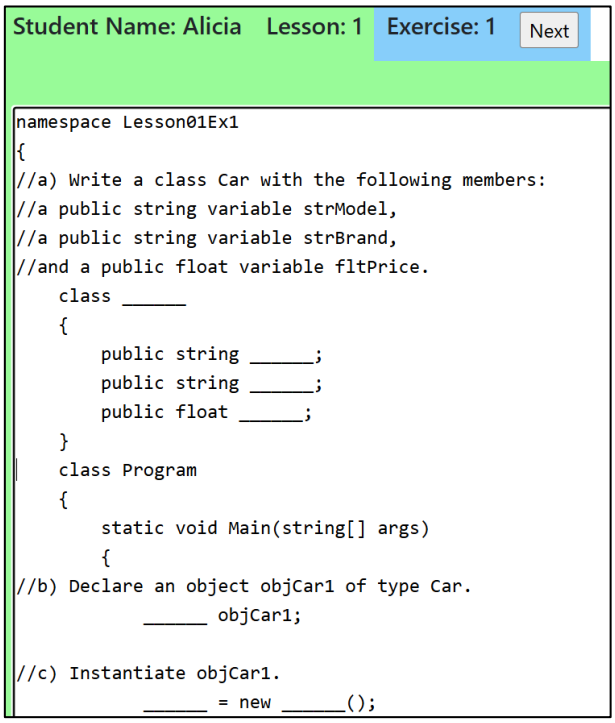

Figure 3 CSHARPY Mode 2

This Guided Learning Mode provide structured support for students through an interactive, **step-by-step approach** to programming instruction. This mode implements a scaffolded learning strategy through the following features:

- When a lesson is selected, CSHARPY displays partially completed code with **blanks** for students to fill in (Figure 3). This template maintains sufficient program structure while leaving key concepts for student completion.

- If a student enters incorrect code or leaves a blank unanswered, CSHARPY automatically provides a **mini-video tutorial** (Figure 4) focusing specifically on the concepts needed for that code segment.

- This just-in-time instructional approach significantly reduces cognitive load compared to traditional methods that require students to absorb lengthy video tutorials before attempting practical exercises. By integrating learning and practice, students engage actively with programming concepts while receiving immediate, relevant support.

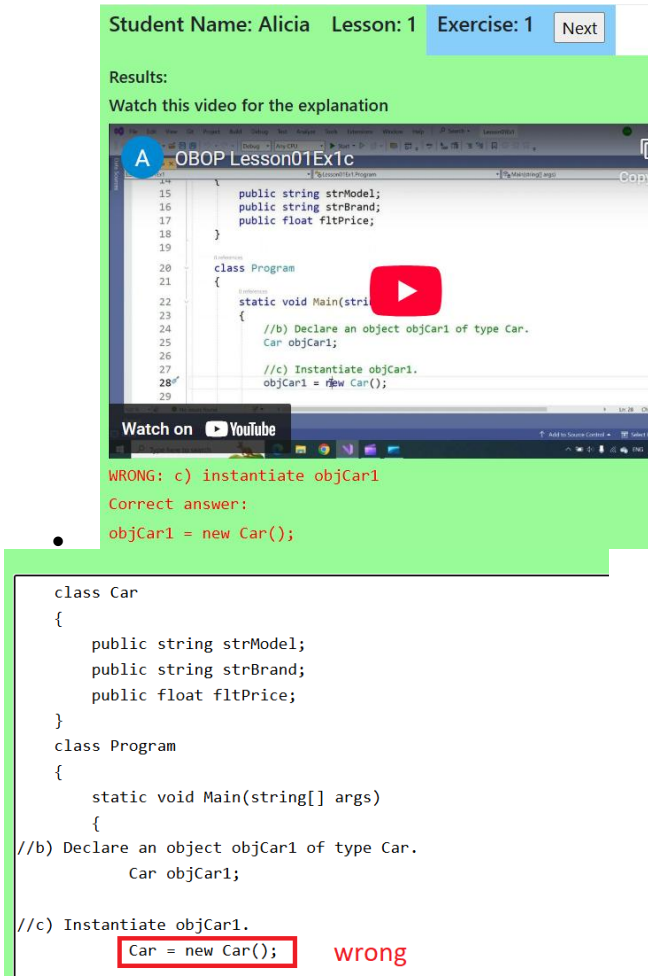

Figure 4 CSharpy mini-video

This scaffolded learning approach facilitates progressive skill development through structured support mechanisms. Rather than emphasising rote memorisation of syntactical elements, the system employs strategic code completion exercises that enable students to develop both structural understanding and logical reasoning skills.

This pedagogical approach aligns closely with contemporary software development practices, where developers primarily work on maintaining, enhancing, and modifying existing code rather than creating new applications from scratch. By emphasising code comprehension and providing contextualised assistance, CSHARPY creates an authentic learning environment that reflects real-world programming practice. This approach particularly benefits novice programmers who might otherwise find traditional programming instruction intimidating, thereby fostering a more accessible and inclusive learning environment.

CSHARPY comprises 15 lessons, each containing one to four scaffolded exercises. This progressive structure allows students to build confidence and understanding through guided practice, starting with fill-in-the-blank tasks before advancing to complete program writing. When ready, student can transition to independent coding using "I have completed the exercise. Please check" mode.

The system's approach reflects real-world software development, where programmers typically modify existing code rather than writing programs from scratch. By emphasising code comprehension and providing targeted assistance, CSHARPY creates a supportive environment that helps students overcome initial programming anxiety while developing practical skills.

Constructivism

Constructivism posits that learners actively construct knowledge through experience and reflection, rather than passively absorbing information. This module embraces a Constructivist learning pedagogy by immersing students in hands-on activities centered around Object Oriented Programming. Instead of relying on traditional lectures, each lesson is designed to engage students in writing, testing, and debugging code—promoting a cycle of experimentation, failure, and refinement that deepens conceptual understanding.

This approach is further enriched by principles drawn from Vygotsky's Zone of Proximal Development (ZPD), which highlights the difference between what a learner can achieve independently and what they can achieve with guidance. Within this framework, learning activities are scaffolded to provide structured support as students take on increasingly complex programming challenges. Initially, guidance is provided through example-driven tasks, collaborative coding exercises, and targeted feedback. As learners build confidence and competence, this scaffolding is gradually removed, encouraging autonomy and mastery. Through this guided, active engagement, students internalize not just the syntax and structure of Object Oriented Programming, but also its underlying principles and practical applications in software development.

Lessons 1 to 3 below illustrate the step-by-step nature of the proramming exercises.

Lesson 1: Introducing Classes and Objects

Lesson 1 introduces students to fundamental Object Oriented Programming concepts as they create their first class. Students learn to define a class and instantiate objects, establishing the foundational understanding of class structure that underpins subsequent lessons.

Lesson 1 programming exercise:

a) Define a class named Car with the following public members: strModel (string), strBrand (string), fltPrice (float) b) Declare an object named objCar1 of type Car. c) Instantiate objCar1. d) Prompt the user to input the model, brand, and

price for objCar1.
e) Display the model, brand, and price of objCar1 on the console.
f) Declare and instantiate a second object named objCar2 of type Car.
g) Prompt the user to input the model, brand, and price for objCar2.
h) Display the model, brand, and price of objCar2 on the console.

Lesson 2: Encapsulation with Properties

Lesson 2 advances students' understanding of class design by introducing data encapsulation. Students learn to protect class data using private members and control access via properties, implementing fundamental Object Oriented principles of information hiding.

Lesson 2 programming exercise:

a) Define a class named Employee with the following members: strName (public string), fltSalary (private float)
b) Implement a public property named Salary that provides controlled access to the private fltSalary variable. The set accessor should only allow values greater than zero to be assigned to fltSalary. The get accessor should return the current value of fltSalary.
c) Declare and instantiate an object named empObj of type Employee.
d) Prompt the user to enter the name and salary for empObj.
e) Display the name and salary of empObj on the console.

Lesson 3: Constructors and Methods

Lesson 3 introduces constructors, methods, and read-only properties through the implementation of a Circle class. Students apply Object Oriented principles while working with mathematical concepts, strengthening their understanding of both domains.

Lesson 3 programming exercise:

a) Define a class named Circle with a private float variable fltRadius.
b) Implement a read-only public property named Area that calculates and returns the area of the circle using the formula: Area=π×fltRadius2 (Use 3.1416f for π).
c) Write a constructor for the Circle class that accepts one float parameter representing the radius and initializes the fltRadius member with this value.
d) Implement a public method named Circumference that calculates and returns the circumference of the circle using the formula: Circumference=2×π×fltRadius (Use 3.1416f for π).
e) Declare two objects, c1 and c2, of type Circle.
f) Prompt the user to enter the radius for both c1 and c2.
g) Instantiate c1 and c2 using the provided radii.
h) Calculate and display the total area of both circles.
i) Calculate and display the total circumference of both circles.

Each exercise demonstrate Object Oriented principles model real-world entities and their behaviour, allowing students to solve real-world problems through code. Building incrementally on previous lessons, students construct their understanding by connecting new concepts to existing knowledge. This constructivist approach fosters deeper comprehension and retention of Object Oriented Programming principles.

Results and Discussion

A CSHARPY user experience survey was taken by 82 students during the October 2024 semester. The survey results are presented in Figures 5 to 8.

Figure 5 Question 1: CSharpy helps me to find and correct mistakes in my programs efficiently

Figure 6 Question 2: CSharpy provides personalised feedback when I am writing my programs

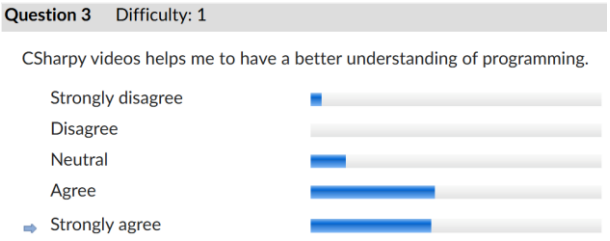

Figure 7 Question 3: CSharpy videos helps me to have a better understanding of programming.

Figure 8 Question 4: I feel more motivated to do my programming exercises when using CSharpy

Question 5: Do you have any comments for the CSharpy Personalised Coding Tutor?

Students' comments:

1	CSharpy is useful towards helping me in my education.
2	I feel that this is very important and effective and will help me in my learning
3	it is very well designed, and particularly helps the students who require a bit more help
4	Its actually genius, I genuinely have nothing bad to say about it
5	the app is well created
6	No comments but i think the CSharpy is really good
7	Well made website that aids in my OAL at home, and I can rely on it to teach me and it is manageable and not difficult to use.
8	You have done a great job to help all students who struggle to finish their code. Thank You!!
9	Really appreciated Csharpy, helped me in my tutorials to better understand where I made syntax or logical errors in my code which would have taken up more time if not for Csharpy.
10	Csharpy and the notes given to us to help us understand the module
11	csharpy was pretty cool and helpful, lesson vids were detailed and helpful
12	nice program which helps students learn the key pointers (CSharpy)
13	The notes and Csharpy helped alot when doing the exercises.

The survey results demonstrated strong student satisfaction with CSHARPY across multiple dimensions:

- **Debugging Support**: 89% of students agreed that CSHARPY helps them identify and correct programming errors effectively.
- **Instructional Guidance**: 84% reported that the integrated video tutorials enhanced their understanding of programming concepts.
- **Targeted Support**: 80% valued the platform's provides personalised feedback on their code submissions.
- **Motivation**: 75% experienced increased motivation to complete programming exercises when using CSHARPY.

Qualitative feedback reinforced CSHARPY's effectiveness as a learning tool. Students particularly valued its intuitive interface, clear instructional videos and comprehensive syntax and logical errors detection capabilities. These comments establish CSHARPY as an essential learning tool and reliable resource for programming education.

Conclusions

CSHARPY demonstrates an innovative and effective solution for teaching Object Oriented Programming in a personalized and adaptive manner. Its dual-mode approach accommodates diverse learning needs, supporting both guided practice and independent learning. The platform's constructivist design, emphasizing on hands-on coding practice and targeted feedback, enables students to progressively develop their competence in Object Oriented Programming concepts, while building confidence.

The positive feedback from students underscores the platform's effectiveness, with many highlighting its usefulness in providing targeted assistance and reinforcing key programming principles through practical exercises. The platform's combination of interactive feedback and hands-on coding exercises addresses traditional challenges in teaching Object Oriented Programming. From October 2024 semester survey results, students reported increased motivation and deeper conceptual understanding, validating CSHARPY's scaffolded learning approach. By effectively supporting learners at different proficiency levels, the system creates an inclusive environment for learning Object Oriented Programming.

In the future CSHARPY will also be used for other programming modules such as the Data Structures and Algorithms module for Year 2 students from Engineering Science Diploma course.

Acknowledgements

The author thanks Mr. Kwek Lian Chong, Senior Lecturer at Ngee Ann Polytechnic for his continuous support in implementing CSHARPY.

The author is also grateful to Mr. Hee Juay Jiunn, Senior Lecturer at Ngee Ann Polytechnic for his support in testing the system.

References

Peters, J., Le Cornu, R. & Collins, J. (2003). Towards Constructivist Teaching and Learning

Bada, S. O. (2015). Constructivism Learning Theory: A Paradigm for Teaching and Learning

Allen, A. (2022). An Introduction to Constructivism: Its Theoretical Roots and Impact on Contemporary Education