A cartoon cat holding a blue box

Description automatically generated

\*\*Maki Studios C# Coding Standards\*\*

---

📜 \*\*Naming Conventions\*\*

\*Guiding Principle:\*

- Aim for clarity. If someone else reads it, they should understand it.

\*Methods:\*

- Use camelCase:

🎯 `void startEngine();`

- Properties start with an uppercase:

🎯 `public int EngineFuel { get; set; }`

\*Variables:\*

- Follow camelCase:

🎯 `int fuelAmount;` or `int timerCounter;`

- Loops may use `i`, `j`, `k`, but aim for descriptive names elsewhere.

- Constants are LOUD with ALL CAPS:

🎯 `const int MAX\_SPEED = 120;`

\*Classes:\*

- Use PascalCase:

🎯 `class RacingCar { ... }`

\*Files:\*

- Begin with your initials (Uppercase) + description:

🎯 `MSRacingCar.cs`

---

✒️ \*\*Commenting\*\*

\*Golden Rule:\*

- If it takes a second to think about, it takes a comment.

- Short annotations:

🎯 `// Initializing engine`

- For deeper insights:

🎯

```

/\*

Initializing engine and checking for

possible overheat scenarios.

\*/

```

---

📏 \*\*Styling and Structure\*\*

\*Indentation:\*

- Stick to 4 spaces. No tabs!

\*Whitespace:\*

- Keep 1-2 lines of spacing between method or class declarations.

\*Brackets:\*

- `{ }` start on a fresh line for classes and methods.

🎯

```

public void IgniteEngine()

{

...

}

```

---

❌ \*\*Handling Errors\*\*

- Leverage C#'s `Debug.Assert(condition, "Descriptive Error Message");`

- Craft meaningful error messages:

🎯 `Debug.Assert(fuelAmount > 0, "Error: Fuel tank is empty!");`

---

🧪 \*\*Testing Principles\*\*

- Create unit tests for essential functions.

- Within test methods, use `Assert.IsTrue(condition, "Descriptive failure message");`

---

📝 \*\*Documentation Habits\*\*

- At the head of each file:

🎯

```

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* RacingCar.cs

\* Jordan Fisher, Maki Studios

\* Manages car racing mechanisms.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

```

- Elaborate on intricate logic in your code:

🎯 `// Adjusting speed based on terrain friction`

---

💡 \*\*General Best Practices\*\*

- Limit method lengths to around 50 lines.

- Organize related methods within regions.

- Strive for code that describes itself over lengthy comments.

---

Drive forward and code smoothly, Maki Studios! 🏎️🏁