Commenting, Style & Ex3 Walkthrough

Software Engineering 1

Dr Swen E. Gaudl

University of Plymouth 2022

COMP10000

Prep For Thursday:

- Try going through Ex4 and come with questions
- Specific CW questions

COMP1000 Agenda This Week:

- Git Practice
- Commenting in C#
- Style Guides for C#
- Exercise 3 Walkthrough
- Q&A

Git Help Session

- Further reading: git-cheat-sheet-education
- Use git scm command line client where possible: git-scm.com

`git clone --recursive[url]`
retrieve an entire repository and its sub-repositories from a hosted location via URL

`git status`
show modified files in working directory, staged for your next commit

`git add [file]`
add a file as it looks now to your next commit (stage)

`git reset [file]`
unstage a file while retaining the changes in working directory

`git diff`
diff of what is changed but not staged

`git commit -m "[descriptive message]"`

`git push [alias] [branch]`
Transmit local branch commits to the remote repository branch

`git pull`
fetch and merge any commits from the tracking remote branch

Commenting & Code Documentation

- 3 types:
  - Multi-Line:

```
/**
* This is a multiline comment to describe or annotate code
* It will be removed when compiled
*/
```

  - Single Line:

```
// some extra comment, also removed when compiled
```

  - Doc Style:

```
/// <summary>
/// A generic Monster Entity for a game.
/// Can be compiled into documentation
/// </summary>
```

Commenting & Code Documentation

- Further reading: programming-guide/xmldoc/
- Comment complex pieces of code
- Mention issues or potential broken points
- Track current work happening by a specific author
- Describe functionality

- Literate Programming (Knuth, paper)
- Why documentation matters (link)

Commenting & Code Documentation

```
/// <summary>
/// Used for receiving damage from other game entities. Reduced health of the Entity.
/// </summary>
/// <param name="damage">Accepts in integer damage value that represents the damage the
/// entity receives.</param>
/// <returns>True if damage could be taken, False otherwise or if the Entity is already
dead.</returns>
public bool TakeDamage(int damage)
{
    if (health > 0)
    {
        health = health - damage;
        return true;
    }

    return false;
}
```

Writing Code

Style Guide:

- ktaranov/naming-convention
- lhunt/CSharp-Coding-Standards
- dofactory.com
- google/csharp-style

```
public class Monster
{
    protected float health = 100.0f;
    protected float damage = 5.0f;
    protected float armor = 100.0f;
    public static int VERSION = 0;

    public bool TakeDamage(int damage)
    {
        float health = health + armor;
    }
    public void DealDamage(Monster[] monsters)
    {}

    public static void Main()
    {
       if (Monster.VERSION > 0)
          Console.WriteLine("New Monster”+
             ” Class found");

     }

}
```

Writing Code

Object Name	Notation	Length	Plural	Prefix	Suffix	Abbreviation	Char Mask	Underscores
Class name	PascalCase	128	No	No	Yes	No	[A-z][0-9]	No
Constructor name	PascalCase	128	No	No	Yes	No	[A-z][0-9]	No
Method name	PascalCase	128	Yes	No	No	No	[A-z][0-9]	No
Method arguments	camelCase	128	Yes	No	No	Yes	[A-z][0-9]	No
Local variables	camelCase	50	Yes	No	No	Yes	[A-z][0-9]	No
Constants name	PascalCase	50	No	No	No	No	[A-z][0-9]	No
Field name	camelCase	50	Yes	No	No	Yes	[A-z][0-9]	Yes
Properties name	PascalCase	50	Yes	No	No	Yes	[A-z][0-9]	No
Delegate name	PascalCase	128	No	No	Yes	Yes	[A-z]	No
Enum type name	PascalCase	128	Yes	No	No	No	[A-z]	No

Exercise 3 Dive…