219114/115 Programming I

OOP programming laboratory 4

Exercise: "Creature Kingdom" – An Inheritance-Based OOP Challenge

You will build a small simulation of creatures in a fantasy world. You will implement a class hierarchy using inheritance and method overriding. The exercise is divided into “features” that align perfectly with Git branching, a concept in version control that we will also be teaching you in this lab.

(main branch)

Create a base class: Creature

Attributes:

- name
- hp (hit points)
- attack_power

Methods:

- attack(target) → prints a generic attack message and reduces target HP
- is_alive() → returns True/False

This is the starter code that all branches inherit from.

https://gist.github.com/parujr/97a1f01e12d328a079a0a03492ab060a

Create a folder and call it **oop_creature_code** and initialize Git:

```
git init

git branch -m master main
```

Create **creature_simulation.py** containing only the base class. Make sure you test the code that it is working correctly. Then:

Commit it:

```
git add creature_simulation.py

git commit -m "Add base Creature class"
```

Feature Branches

Branch 1: flying_creature

```
git checkout -b flying_creature
```

Create a subclass: FlyingCreature(Creature)

Extra attributes:

- altitude (default 0)

Extra methods:

- fly_to(new_altitude)
- Override attack() so flying creatures attack from above.

Fill in the part for class FlyingCreature(Creature):

https://gist.github.com/parujr/2e245be5bf8fbf96d68a770a8820ce1b

The correct code should give the following result:

https://gist.github.com/parujr/172a00dffcfe2939004450b5c94e32a1

After you are done:

```
git add creature_simulation.py
```

```
git commit -m "Add FlyingCreature"
```

To see how your repo looks like at the moment:

```
git log --all --graph
```

Branch 2: swimming_creature

```
git checkout main
```

```
git checkout -b swimming_creature
```

Create a subclass: SwimmingCreature(Creature)

Extra attributes:

- depth (default 0)

Extra methods:
- dive_to(new_depth)
- Override attack() so they attack differently underwater.

Fill in the part for class SwimmingCreature(Creature):

https://gist.github.com/parujr/8147af14a7f63f11c273519d8c8f5b55

The correct code should give the following result:

https://gist.github.com/parujr/d7d81700dfc9c9044ba94e3eb09ae944

After you are done:

```
git add creature_simulation.py
```

```
git commit -m "Add SwimmingCreature"
```

To see how your repo looks like at the moment:

```
git log --all --graph
```

Merging

Let's now merge branches into main:

```
git checkout main
```

```
git merge flying_creature
```

To see how your repo looks like at the moment:

```
git log --all --graph
```

Then:

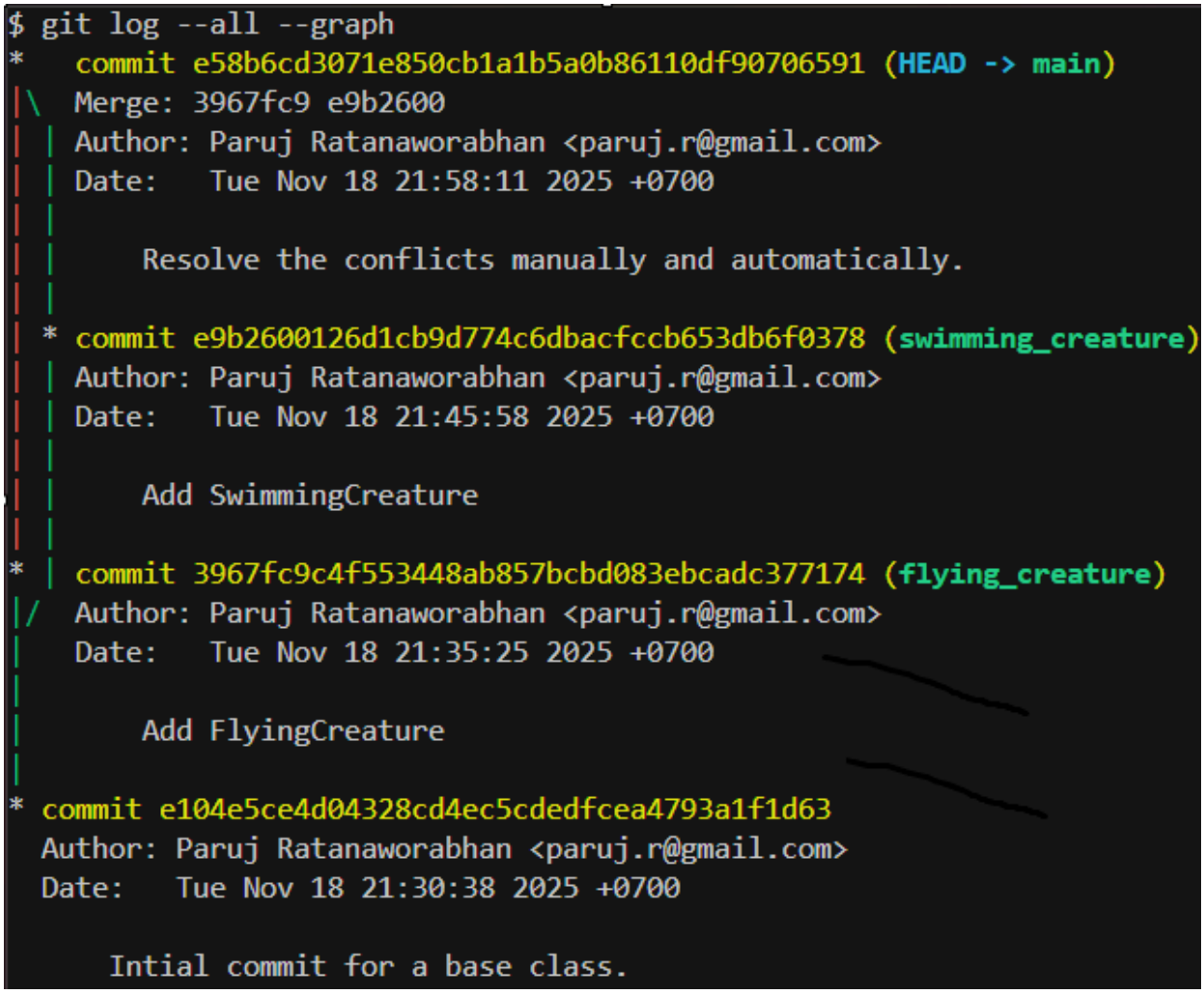
```
git merge swimming_creature
```

This produces merge conflicts that must be resolved. Once done, you should commit your change:

```
git commit -m "Resolve the merge conflict successfully."
```

Your repo should look like the following with

`git log --all --graph`



Branch 3: fire_creature

`git checkout main`

`git checkout -b fire_creature`

Create a subclass: FireCreature(Creature)

Extra attributes:

- fire_level (0–100)

Extra methods:
- emit_fire(new_fire_level)
- Override attack() to incorporate fire.

Then, create a test code for this fire_creature.

Once done, merge this branch to the main branch. Then, create a README.md file using the following template.

Creature Battle Project (OOP + Git Branching Practice)

This project is designed to help me learn **object-oriented programming** in Python — especially **inheritance** — while also practicing key **Git concepts** like branching, merging, conflict resolution, and version history.

I will begin with a base class (`Creature`) and create additional subclasses on separate Git branches to simulate a real development workflow.

Learning Objectives

Python / OOP
- Understand and implement **classes**
- Use **inheritance** to extend behaviors
- Override and extend methods (e.g., `attack()`)
- Practice writing simple test code in `main`

Git / Version Control
- Create and switch branches
- Merge branches into `main`
- Resolve merge conflicts
- View commit history (`git log --graph --all`)

Project Structure
oop_creature_code/
│
├── creature_simulation.py # Base class + subclasses
└── README.md

Project Structure
Describe how to test and run your program

Add the README.md file to your local repo and, finally, push your repo to your remote Github repo.

What to submit

- Show your final commit at your remote Github repo. to the TAs so they can check you off
- Put the link to the Github repository to Google Classroom for grading