Writing Assignment Discussion / Program Design Activity

CSCI-3081: Program Design and Development

Duck example adapted from Head First Design Patterns by Freeman and Robson

5-minute “start the class off right” writing exercise

- Prompt: Spend the next 5 minutes formulating an unresolved question that you would like to ask the authors about the software development writing that you read.

UNIVERSITY OF MINNESOTA
PROFESSOR DANIEL F. KEEFE

Type	Audience	Purpose
Quick Start Guide	Programmers with some general experience but not with this particular software.	Learning a new topic, act as a reference
Research Article	Experts in the subject	Discussion new concept, Share information, Present results, Advance the field
API Documentation	Developers with more experience who want to use this API	Make it easier to use API, supplementing code. Describe interface (but not implementation)
Datasheet	EE or CE developers near to hardware	Familiarize; quick reference
Article — trade magazine, online magazine	Project manager	Design advice
Programmer's blog	Ranges — intro level to expert programmers	Philosophical arguments, career advice, real-world example

1. States key information clearly (important findings, recommendations, ideas, issues, etc.).
2. Addresses the target audience with an appropriate tone and level of explanation.
3. Presents any needed background explanation clearly or informatively.
4. Persuasively justifies any choice of algorithm, design, problem approach, problem solution, etc.
5. Describes algorithms, data structures, software system components, etc. accurately.
6. Describes algorithms, data structures, software system components, etc. informatively and concisely.
7. Uses terminology and notation correctly.
8. Explains high-level ideas with low-level details.
9. Presents low-level details clearly and accurately.
10. Uses appropriate structures (e.g., lists, diagrams, equations, code samples, etc.).
11. Tables, diagrams, references, etc. are clear and informative.
12. Tables, diagrams, references, etc. are smoothly integrated into the text.
13. Has a good organization and logical flow.
14. Uses correct grammar, spelling, and mechanics.
15. Avoids including irrelevant information, overemphasizing less important material, or writing verbosely.

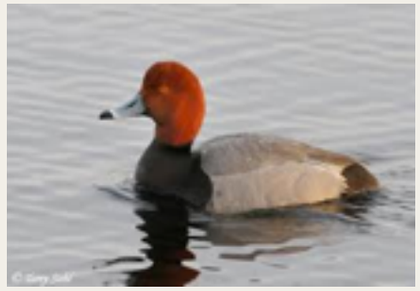
Program Design Activity (Quack, Quack, …)

UNIVERSITY OF MINNESOTA
PROFESSOR DANIEL F. KEEFE

You try an example: Ducks

Mallard

,

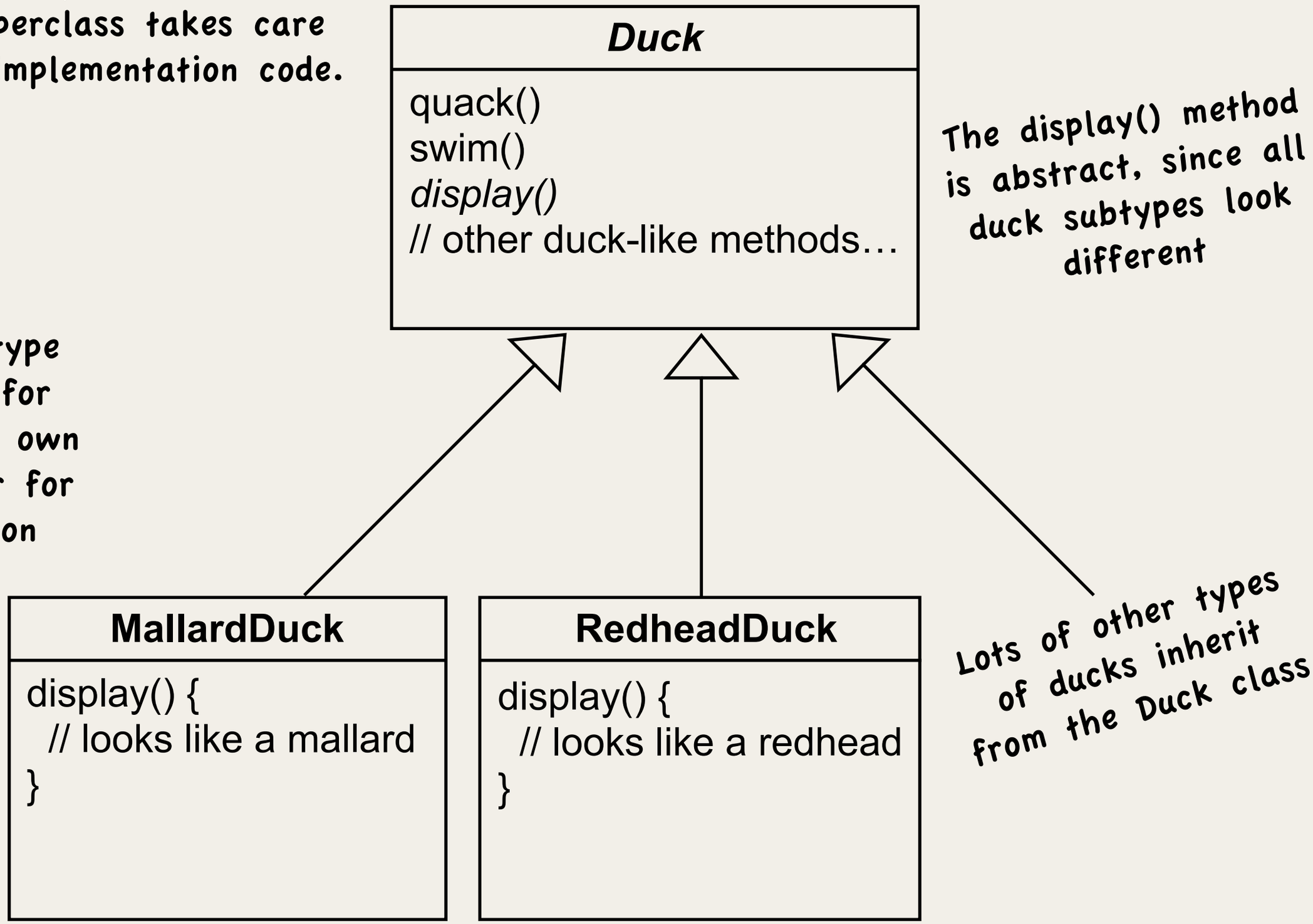


RedHead

, . . .

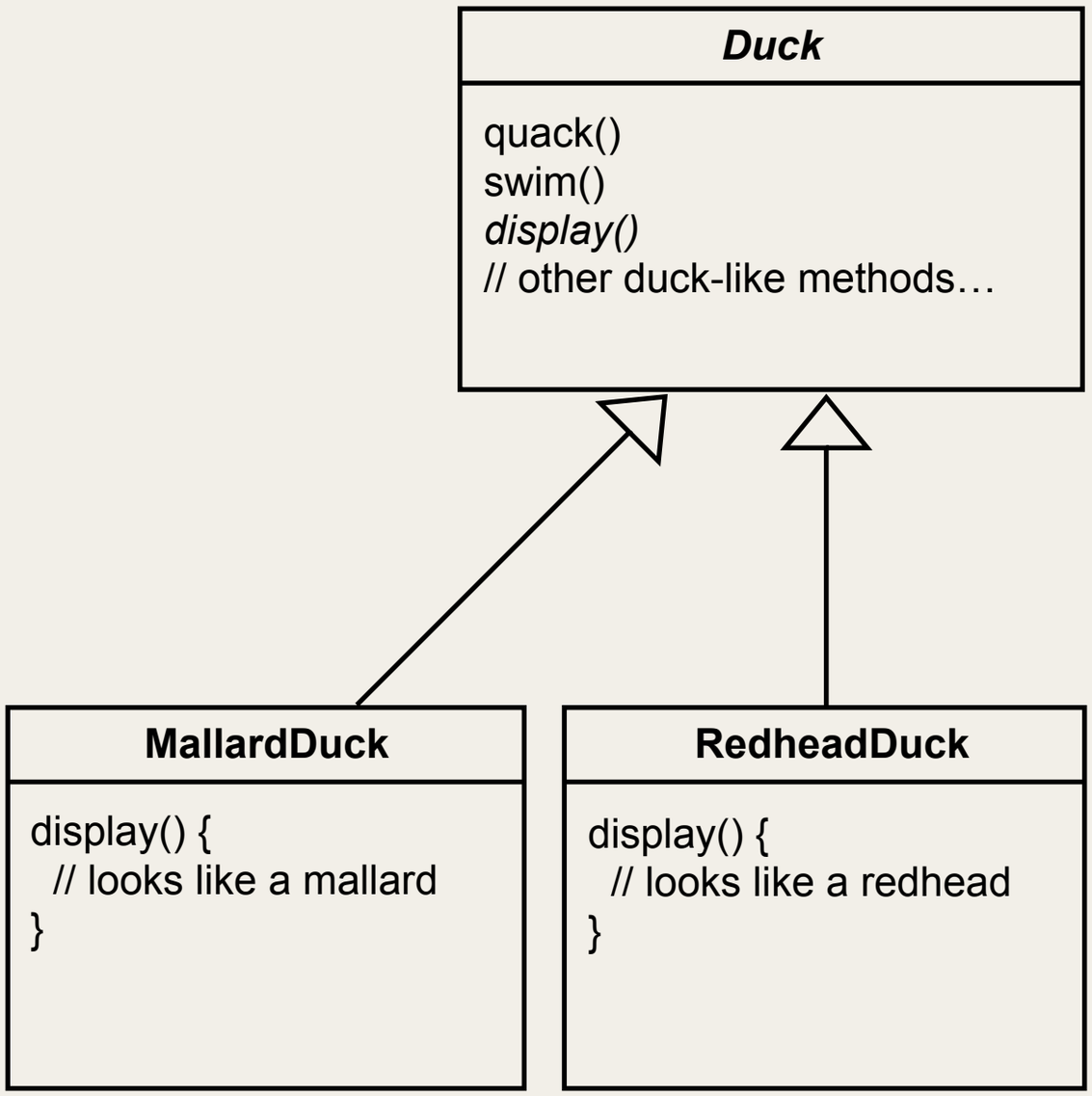
- All ducks **quack()** and **swim()**, the superclass takes care of the implementation details.
- Each duck subtype is responsible for implementing its own **display()** method, since they all look different when drawn on the screen.

All ducks quack and swim, the superclass takes care of the implementation code.

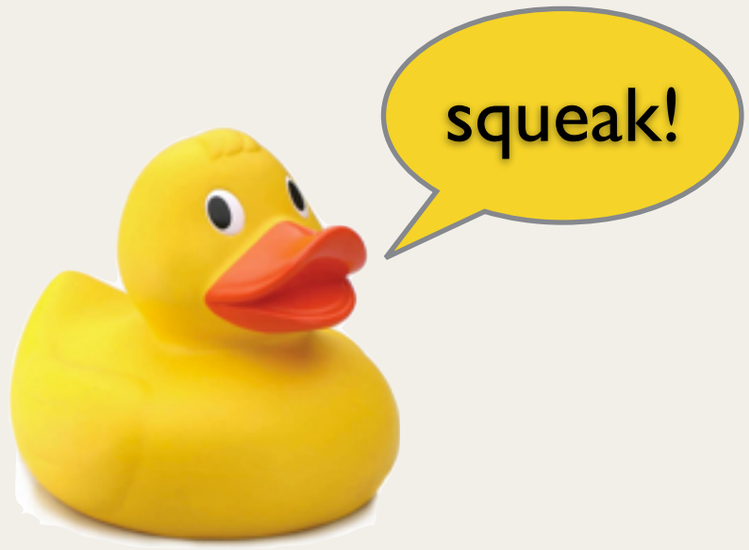


Your First Challenge (this one is easy)

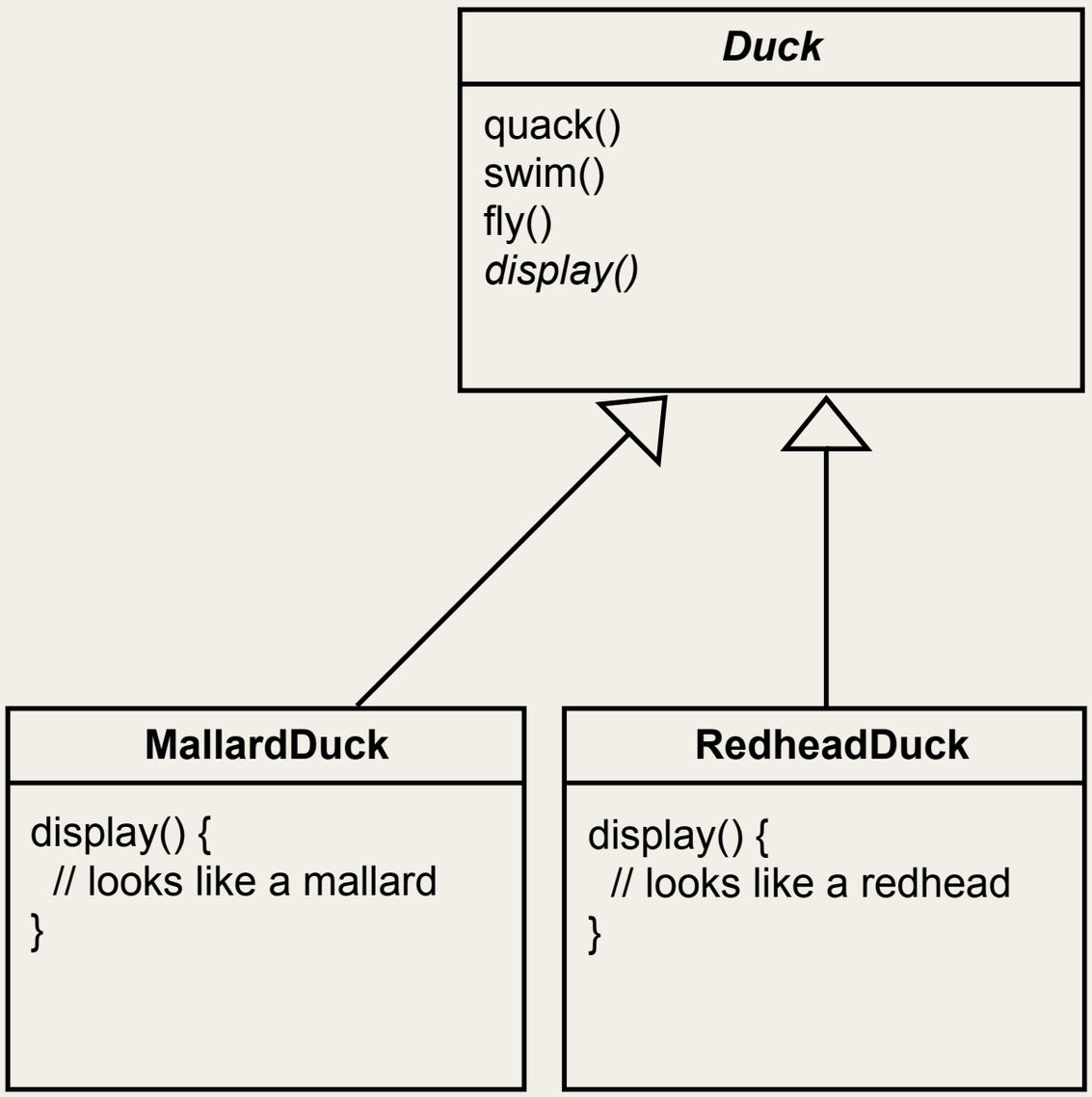
- Extend the program to make Ducks fly



Your Second Challenge

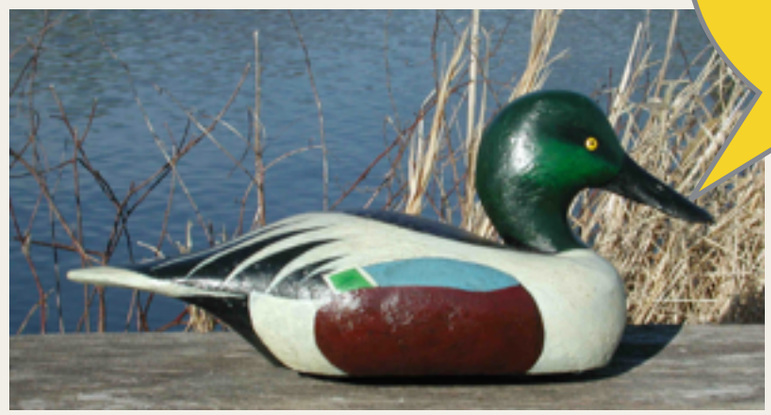


- Add a new type of duck - RubberDuck



Your Third Challenge

- Add a new type of duck - DecoyDuck



Duck

quack()
swim()
fly()
display()

MallardDuck

```
display() {
  // looks like a mallard
}
```

RedheadDuck

```
display() {
  // looks like a redhead
}
```

RubberDuck

```
display() {
  // looks like a rubber ducky
}
quack() {
  // overridden to squeak
}
fly() {
  // overridden to do nothing
}
```

DecoyDuck???

Should we follow the same strategy?

(By the way, we just got a memo saying the executives want to update the product every 6 months!)

to be continued…

UNIVERSITY OF MINNESOTA
PROFESSOR DANIEL F. KEEFE