

Polymorphism



Colorado State University
Department of Computer Science

Slides Originally Created by Marcia Moraes (marcia.moraes@colostate.edu)

Announcements

TODO Reminders:

Readings are due **before** lecture

- Reading 20 (zybooks) – you should have already done that ☺
- Lab 13
- Reading 21 (zyBooks)
- Lab 12
- Reading 22 (zybooks)
- RPA 10

Keep practicing your RPAs in a spaced and mixed manner ☺



<https://www.pinterest.com/pin/342062534189160703/>

Help Desk

| Day | Time : Room |
|-----------|-----------------------|
| Monday | 2 PM - 5 PM : CSB 120 |
| Tuesday | 6 PM - 8 PM : Teams |
| Wednesday | 3 PM - 5 PM : CSB 120 |
| Thursday | 6 PM - 8 PM : Teams |
| Friday | 3 PM - 5 PM : CSB 120 |
| Saturday | 12 PM - 4 PM : Teams |
| Sunday | 12 PM - 4 PM : Teams |

Recall Activity

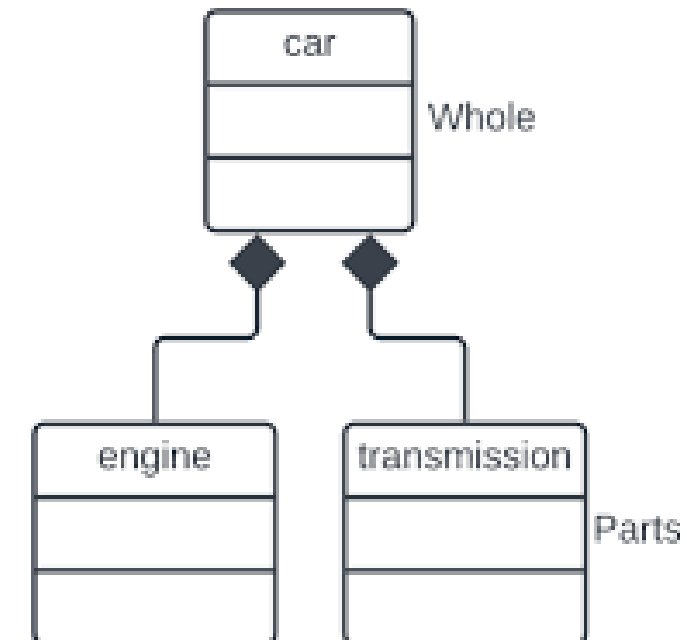
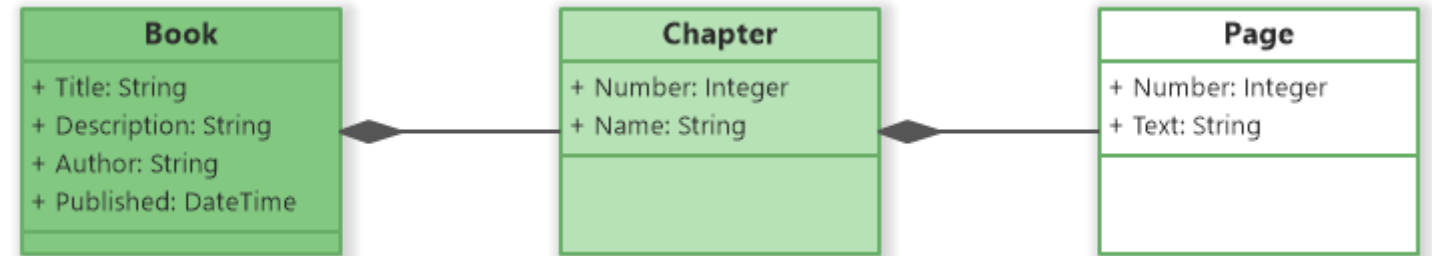
- Analyze the classes below. What type of relationship do we have: 'has-a' or 'is-a'?
- Explain using your own words.

```
public class ChildInfo {  
    public String firstName;  
    public String birthDate;  
    public String schoolName;  
  
    ...  
}  
  
public class MotherInfo {  
    public String firstName;  
    public String birthDate;  
    public String spouseName;  
    public ArrayList<ChildInfo> childrenData;  
  
    ...  
}
```

Composition

- Has-a relationship

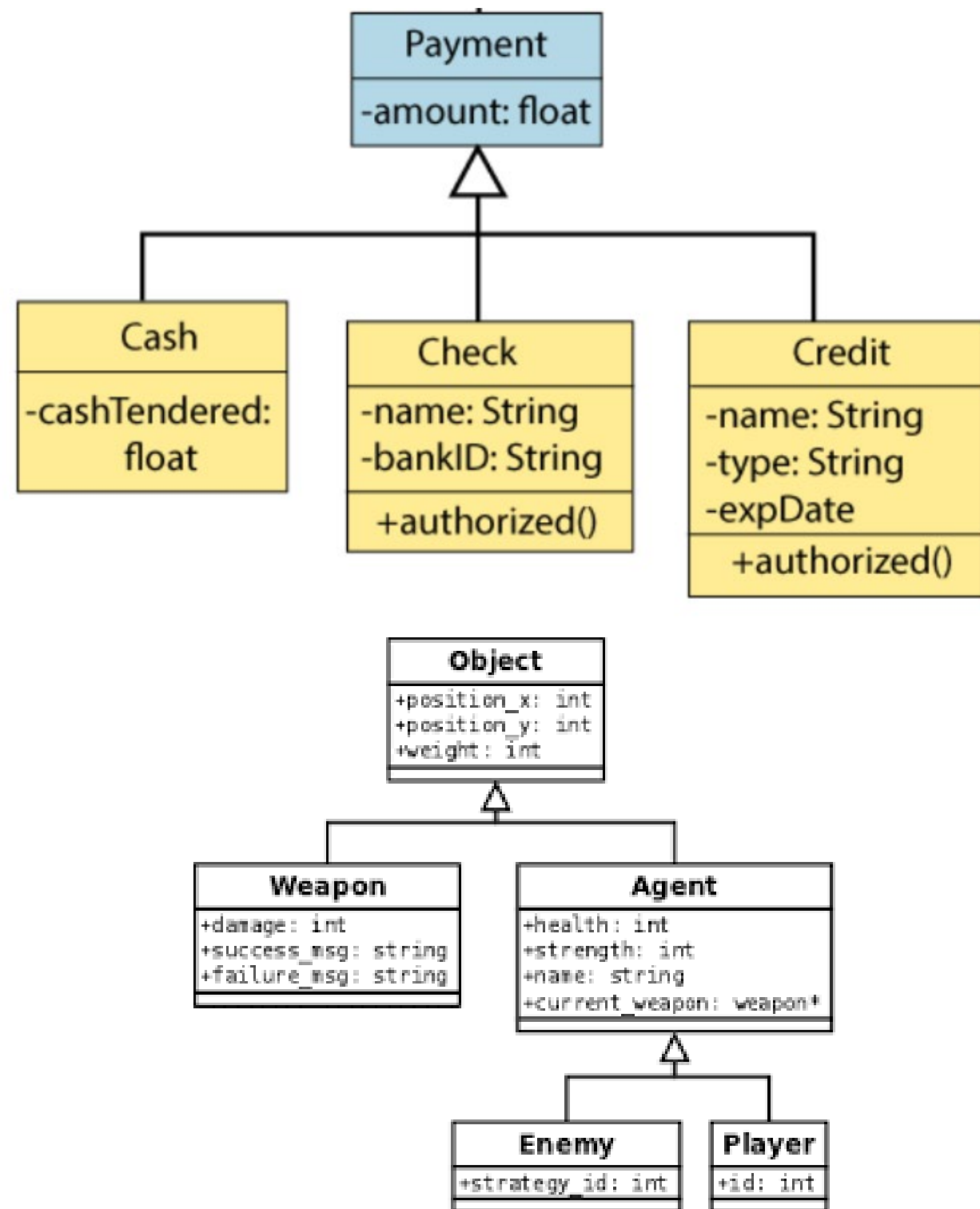
```
public class ChildInfo {  
    public String firstName;  
    public String birthDate;  
    public String schoolName;  
  
    ...  
}  
  
public class MotherInfo {  
    public String firstName;  
    public String birthDate;  
    public String spouseName;  
    public ArrayList<ChildInfo> childrenData;  
  
    ...  
}
```



Inheritance

- Is-a relationship

```
public class PersonInfo {  
    public String firstName;  
    public String birthdate;  
  
    ...  
}  
  
public class ChildInfo extends PersonInfo {  
    public String schoolName;  
  
    ...  
}  
  
public class MotherInfo extends PersonInfo {  
    public String spousesname;  
    public ArrayList<ChildInfo> childrenData;  
  
    ...  
}
```



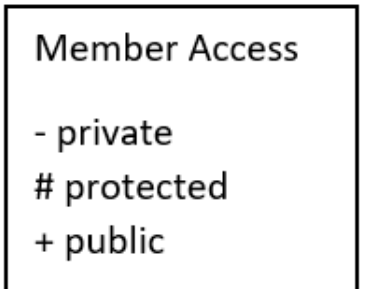
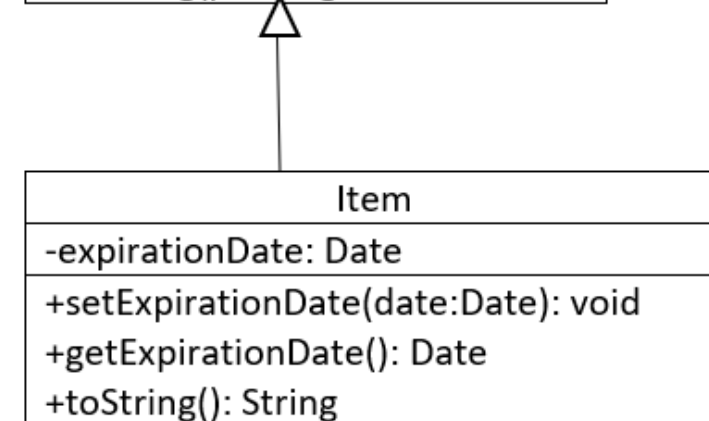
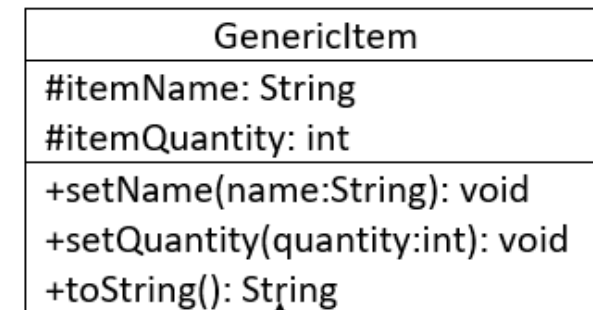
Polymorphism

- Refers to determining which program behavior to execute depending on data types
- Polymorphism of methods – methods overloading
 - **compile-time polymorphism**
 - compiler determines which of several identically-named methods to call based on the method's arguments
- Polymorphism of variables – involves derived classes (inheritance)
 - **runtime polymorphism**
 - compiler cannot make the determination but instead the determination is made while the program is running

Polymorphism of variable

- **Substitution principle** - you can always use a subclass object when a superclass object is expected
- Super class variable can store super class types and sub class types as well
- Sub class variable can only store sub class types

```
GenericItem g1 = new GenericItem();    Correct!  
GenericItem g2 = new Item();           Correct!  
  
Item g3 = new Item();                  Correct!  
Item g4 = new GenericItem();           Incorrect!
```



ArrayList of Objects

- Store a collection of objects of various class types

```
public class Business {  
    protected String name;  
    protected String address;  
  
    public Business() {}  
  
    public Business(String busName, String busAddress) {  
        name = busName;  
        address = busAddress;  
    }  
  
    @Override  
    public String toString() {  
        return name + " -- " + address;  
    }  
}
```

```
import java.util.ArrayList;  
  
public class ArrayListPrinter {  
  
    // Method prints an ArrayList of Objects  
    public static void printArrayList(ArrayList<Object> objList) {  
        int i;  
  
        for (i = 0; i < objList.size(); ++i) {  
            System.out.println(objList.get(i));  
        }  
    }  
  
    public static void main(String[] args) {  
        ArrayList<Object> objList = new ArrayList<Object>();  
  
        // Add new instances of various classes to objList  
        objList.add(new Object());  
        objList.add(12);  
        objList.add(3.14);  
        objList.add(new String("Hello!"));  
        objList.add(new Business("ACME", "5 Main St"));  
  
        // Print list of Objects  
        printArrayList(objList);  
    }  
}
```

```
java.lang.Object@4517d9a3  
12  
3.14  
Hello!  
ACME -- 5 Main St
```


instanceof

```
public static void printArrayListV2(ArrayList<Object> objList) {  
    int i;  
    for (i = 0; i < objList.size(); ++i) {  
        Object obj = objList.get(i);  
        if(obj instanceof String)  
            System.out.println("String:" + objList.get(i));  
        else if(obj instanceof Integer)  
            System.out.println("Integer:" + objList.get(i));  
        else if(obj instanceof Double)  
            System.out.println("Double:" + objList.get(i));  
        else if(obj instanceof Business)  
            System.out.println("Business:" + objList.get(i));  
    }  
}
```

Worksheet

- Complete the Polymorphism worksheet
- Codes from this lecture and worksheet - <https://github.com/CSU-CompSci-CS163-4/Handouts/tree/main/ClassExamples/10Polymorphism>