CSE2016 Programming Methodology

Week 9: Interface

Instructor: Jinyoung Han (jinyounghan@hanyang.ac.kr)

HANYANG UNIVERSITY

Contents

HANYANG UNIVERSITY

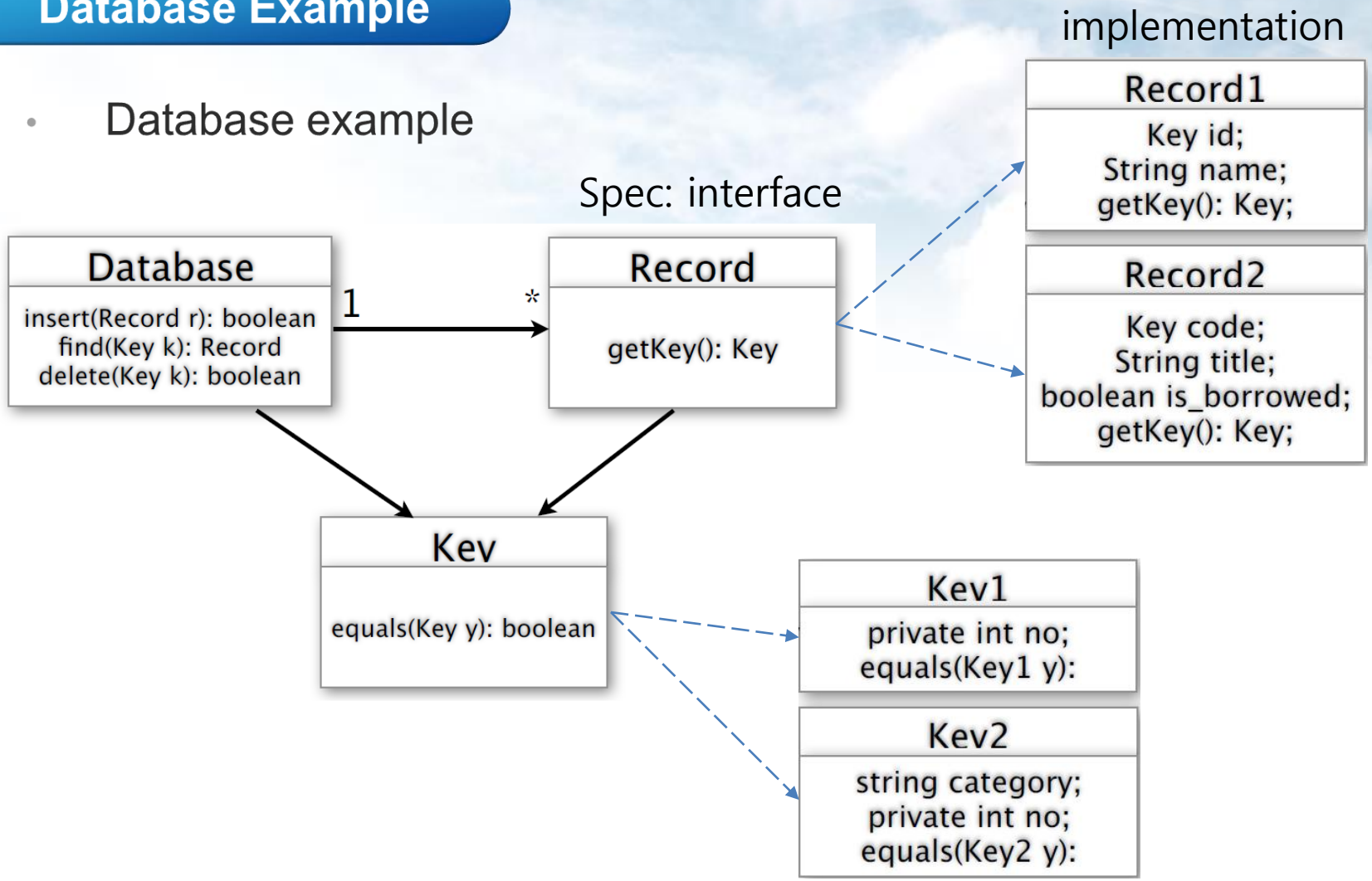
Today's Schedule

1. Java Interfaces
2. Adventure Game
3. Type, Abstract, Package

HANYANG UNIVERSITY

Database Example

- Database example



01. Java Interfaces

HANYANG UNIVERSITY

Interface

- Definition
  - interface <name> { body..}
  - No field initialization
  - Method definition: no implementation, just type and name

- Example
  - interface Key { boolean equals(Key y); }
  - interface Record { Key getKey(); }

Interface as a Type

- Interface can be used as a type, like class
  - No “new”! – no implementation is provided

- Example
  - Variable declaration, Key k;
  - Argument passing, void method (Key K) { … }
  - Return, Key getKey() { … }

HANYANG UNIVERSITY

Interface Implementation

- Interface implementation
  - Using “implements”

```
public class IntegerKey implements Key
{
    private int id;

    public IntegerKey(int j) { id = j;}
    public int getInt() { return id; }

    public boolean equals(Key another_key)
    {
        int m = ((IntegerKey)another_key).getInt();
        return (id == m);
    }
}
```

HANYANG UNIVERSITY

Interface Implementation

- Another implementation

```
public class CodeKey implements Key
{
    private String category;
    private int code;
    public CodeKey(String s, int i) { category=s; code=i; }
    public String getCategory() { return category; }
    public int getCode() { return code; }

    public boolean equals(Key k)
    {
        CodeKey ck = (CodeKey) k;
        return category.equals(ck.getCategory()) && code ==
        ck.getCode();
    }
}
```

HANYANG UNIVERSITY

OOP?

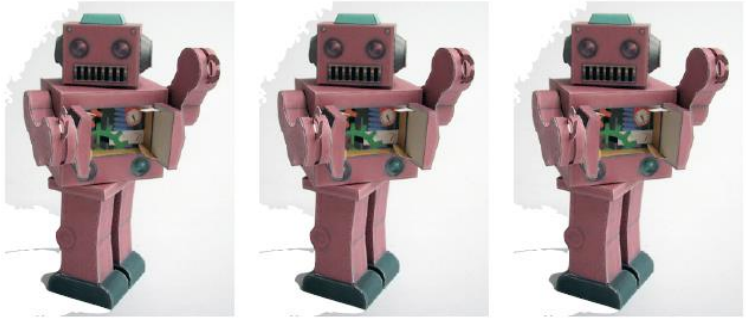
- Class, Object, Inheritance, Interface, Implementation??

Robot
- Conversation
- Moving

Interface

Class

Objects



01. Java Interfaces

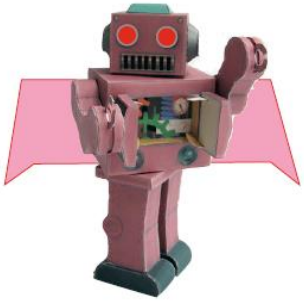
HANYANG UNIVERSITY

OOP

- Inheritance: reusing implementation

Class A

Class B extends A



Basic function+
Laser eyes +
Wing

01. Java Interfaces

HANYANG UNIVERSITY

OOP

- Interface implementation: satisfying only requirements

Robot
- Conversation
- Moving

Interface

A Summary

- Similarity: Inheritance vs. Interface implementation
  - class B extends A
  - class B implements A
  - new B();

- Note: differences
  - Inheritance: re-using codes
  - Interface implementation: implementing codes satisfying requirements

Adventure Game

- Game concept
  - Players are exploring rooms
  - When entering a room, a password is needed
- Room
  - A player can enter and leave a room
  - Occupancy info of a room should be provided
- Player
  - can speak
  - can explore rooms

Interface

```
public interface RoomBehaviour {
    public boolean enter(PlayerBehaviour p);
    public void exit(PlayerBehaviour p);
    public PlayerBehaviour occupantOf();
}

public interface PlayerBehaviour {
    public String speak();
    public boolean explore(RoomBehaviour r);
}
```

Basic Room

```
public class BasicRoom implements RoomBehaviour {
    private PlayerBehaviour occupant;
    private String rooms_name;
    private String secret_word;

    public BasicRoom(String name, String password) {
        occupant = null; rooms_name = name; secret_word = password;
    }
    public boolean enter(PlayerBehaviour p) {
        boolean result = false;
        if(occupant == null && secret_word.equals(p.speak())) {
            occupant = p; result = true;
        }
        return result;
    }
    public void exit(PlayerBehaviour p) {
        if(occupant == p) occupant = null;
    }
    public PlayerBehaviour occupantOf() { return occupant; }
}
```

Explorer

```
public class Explorer implements PlayerBehaviour {
    private String name, secret_word;
    private RoomBehaviour where_I_am;

    public Explorer(String n, String w) {
        name = n; secret_word = w; where_I_am = null;
    }
    public String speak() { return secret_word; }
    public void exitRoom() {
        if(where_I_am != null) {
            where_I_am.exit(this);
            where_I_am = null;
        }
    }
    public boolean explore(RoomBehaviour r) {
        if(where_I_am != null) exitRoom();
        boolean went_inside = r.enter(this);
        if(went_inside) where_I_am = r;
        return went_inside;
  }
    public RoomBehaviour locationOf() { return where_I_am; }
}
```

HANYANG UNIVERSITY

Execution Code

```
RoomBehaviour[] ground_floor = new RoomBehaviour[4];
ground_floor[0] = new BasicRoom("kitchen", "pasta");
ground_floor[3] = new BasicRoom("lounge", "swordfish");
Explorer harpo = new Explorer("Harpo Marx", "swordfish");
Explorer chico = new Explorer("Chico Marx", "tomato");
boolean success = harpo.explore(ground_floor[3]);
```

More Interfaces

- If I want to have..
  - A room with treasure
  - A method for obtaining treasure

```
public interface TreasureProperty {
    public String contentsOf();
}
public interface TreasuryRoomBehaviour {
    public boolean enter(PlayerBehaviour p);
    public void exit(PlayerBehaviour p);
    public TreasureProperty yieldTreasure(PlayerBehaviour p);
    public PlayerBehaviour occupantOf();
}
```

HANYANG UNIVERSITY

More Interfaces

- 1st way: implementing interfaces together

```
public interface RoomBehaviour {
    public boolean enter(PlayerBehaviour p);
    public void exit(PlayerBehaviour p);
    public PlayerBehaviour occupantOf();
}

public interface Treasury {
    public TreasureProperty yieldTreasure(PlayerBehaviour p);
}

public class VaultRoom implements Treasury, RoomBehaviour {
    // 방의 속성과 보물이 있는 속성을 동시에 만족하는 코드
    ...
}
```

One class implements multiple interfaces

HANYANG UNIVERSITY

More Interfaces

- 2nd way: extending interface

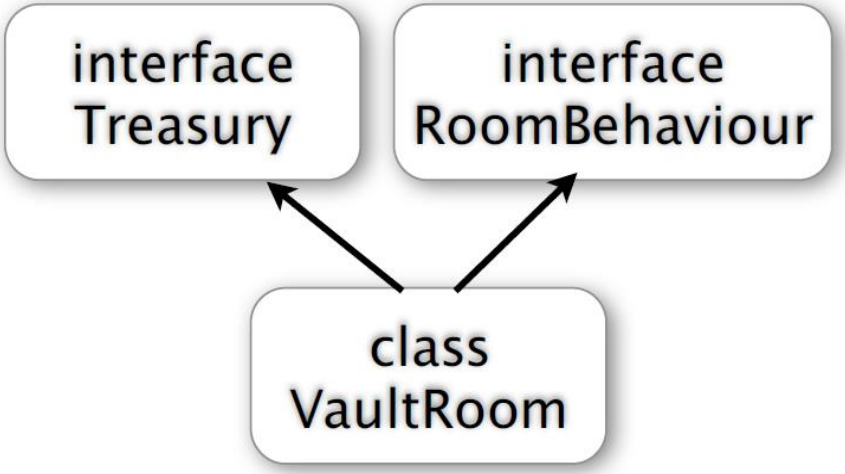
```
public interface RoomBehaviour {
    public boolean enter(PlayerBehaviour p);
    public void exit(PlayerBehaviour p);
    public PlayerBehaviour occupantOf();
}
```

An interface can be extended

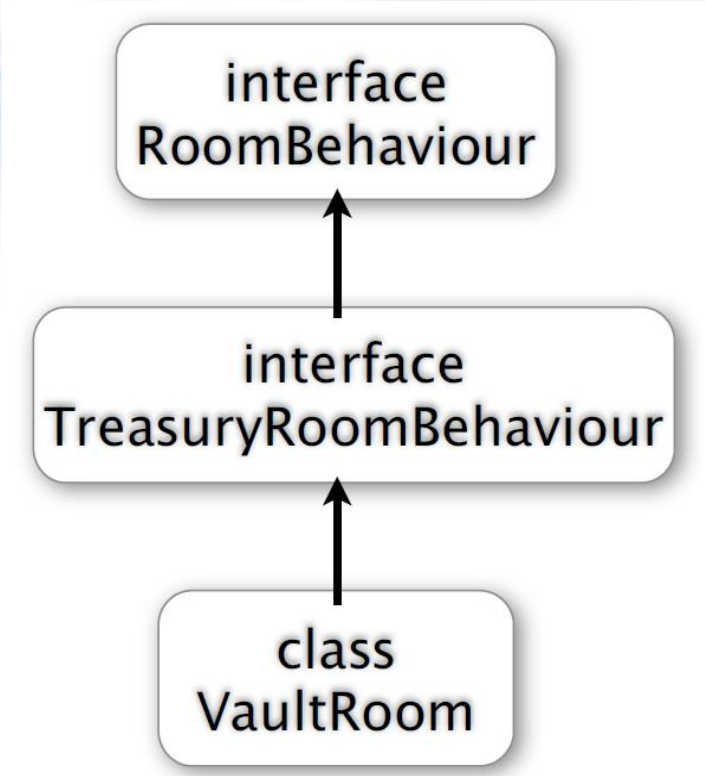
```
public interface TreasuryRoomBehaviour extends RoomBehaviour {
    public TreasureProperty yieldTreasure(PlayerBehaviour p);
}

public class VaultRoom implements TreasuryRoomBehaviour {
    // 보물 있는 방의 속성을 만족하는 코드
    ...
}
```

HANYANG UNIVERSITY

Comparison



[1st way]



[2nd way]

HANYANG UNIVERSITY

Type Checking

- A safe way of “type checking”
  - “instanceof”: testing whether the object is an instance of the specified type

```
class Simple1{
     public static void main(String args[])
     {
          Simple1 s=new Simple1();
          System.out.println(s instanceof Simple1); //true
     }
}
```

```
interface Key {
    boolean equals(Key y);
}

class IntegerKey implements Key {
    ...
    boolean equals(Key k) {
        if(k instanceof IntegerKey)
            return key == ((IntegerKey)k).getInt();
        else
            return false;
    }
    ...
}
```

Abstract

- A class where some methods are not defined
- Good when method implementations are shared
- An example: card player

```
public abstract class CardPlayer implements CardPlayerBehaviour
{
    private Card[] my_hand;
    private int card_count;

    public CardPlayer(int max_cards) {
        my_hand = new Card[max_cards];
        card_count = 0;
    }
    public abstract boolean wantsACard();
    public void receiveCard(Card c) {
        my_hand[card_count] = c;
        card_count = card_count + 1;
    }
}
```

HANYANG UNIVERSITY

Abstract

- Human player

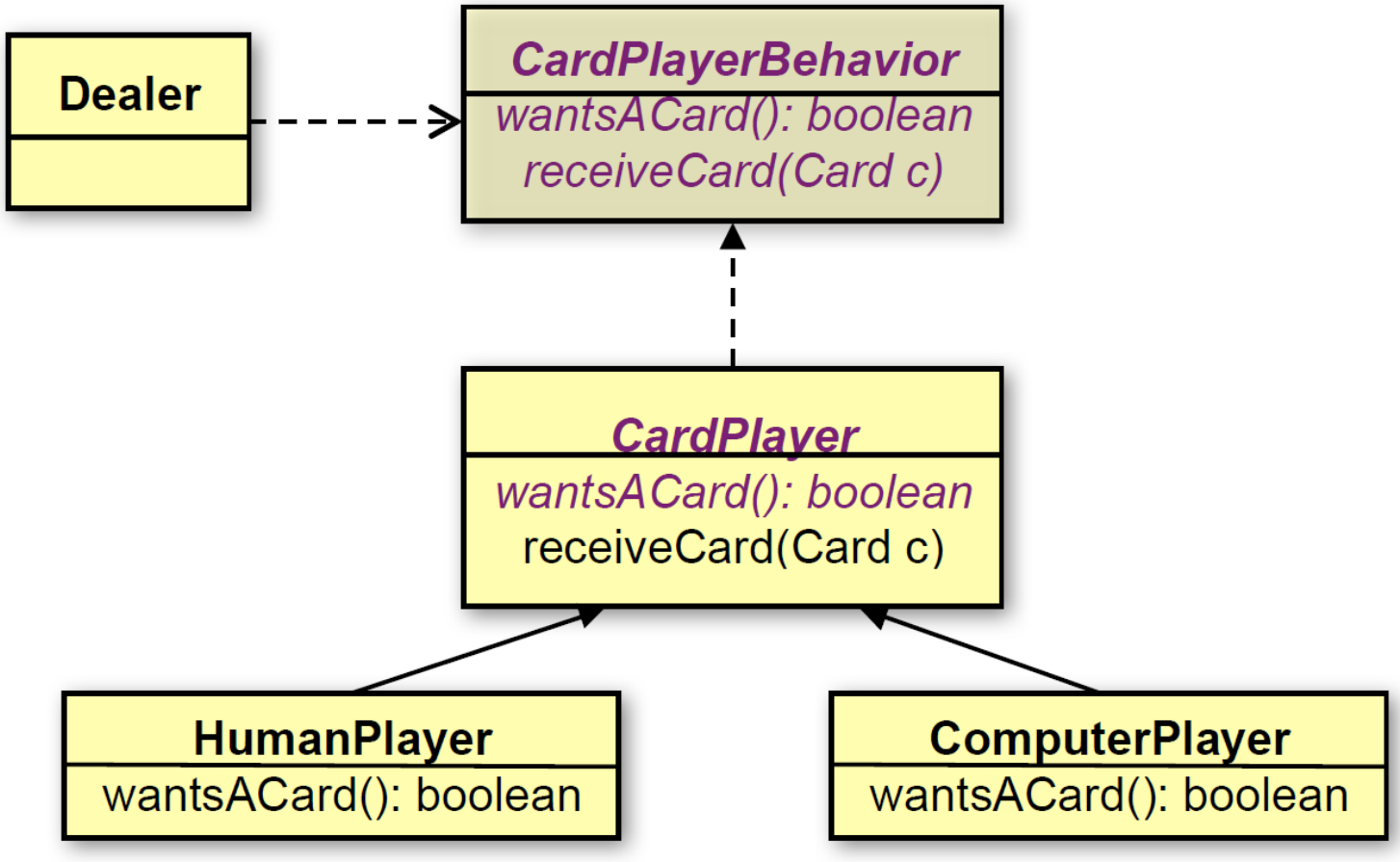
```
public class HumanPlayer extends CardPlayer {
    public HumanPlayer(int max_cards) {
        super(max_cards);
    }
    public boolean wantsACard() {
        String response = JOptionPane.showInputDialog
            ("Do you want another card (Y or N)?");
        return response.equals("Y");
    }
```

- Computer player

```
public class ComputerPlayer extends CardPlayer {
    public ComputerPlayer(int max_cards) {
        super(max_cards);
    }
    public boolean wantsACard() {
        int sum = 0;
        for(int i=0; i<card_count; i++)
            sum += my_hand[i].getCount();
        return sum < 15;
    }
```

Abstract

- Architecture



HANYANG UNIVERSITY

Abstract

- Another example: animal
  - Animal
    - Warm blooded
      - Equine
        - Horse

```
public abstract class Animal
{ // 동물에 관계된 필드와 메소드 }

public abstract class WarmBlooded
                extends Animal
{ // 온혈동물에 관계된 필드와 메소드 }

public abstract class Equine
                extends WarmBlooded
{ // 말과에 관계된 필드와 메소드 }

public class Horse extends Equine
{ // 말에 관계된 메소드를 채움으로써 완성 }
```

Abstract

- Interface vs. abstract class

Interface	Abstract class
Define specification	Partially implemented, partially not
Interface implementation: code satisfying requirements	Reusing inherited codes, adding unimplemented part
Do not care how a class is implemented	If there are multiple classes who share a part of codes which are commonly shared

Package

- Package: a folder including classes, interfaces, etc.
  - E.g., java.util, java.awt, javax.swing
- Using import

Summary

- OOP concepts
  - class, object
  - inheritance
  - overriding, overloading
  - interface, implements
  - abstract
  - package

Thanks

Week 9: Interface

Instructor: Jinyoung Han (jinyounghan@hanyang.ac.kr)