인터페이스와
클래스 계층 구조

Interfaces and Class Hierarchy

슬라이드 퍼즐 게임

Starter

PuzzleStarter

`static void main(String[] args)`

Controller

PuzzleController

`SlidePuzzleBoard board`
`PuzzleWriter writer`

`PuzzleController(SlidePuzzleBoard b, PuzzleWriter w)`
`void play()`

Model

SlidePuzzleBoard

`PuzzlePiece[][] board`
`int empty_row`
`int empty_col`

`SlidePuzzleBoard()`
`boolean move(int w)`

1 *

PuzzlePiece

`int face`

`PuzzlePiece(int n)`

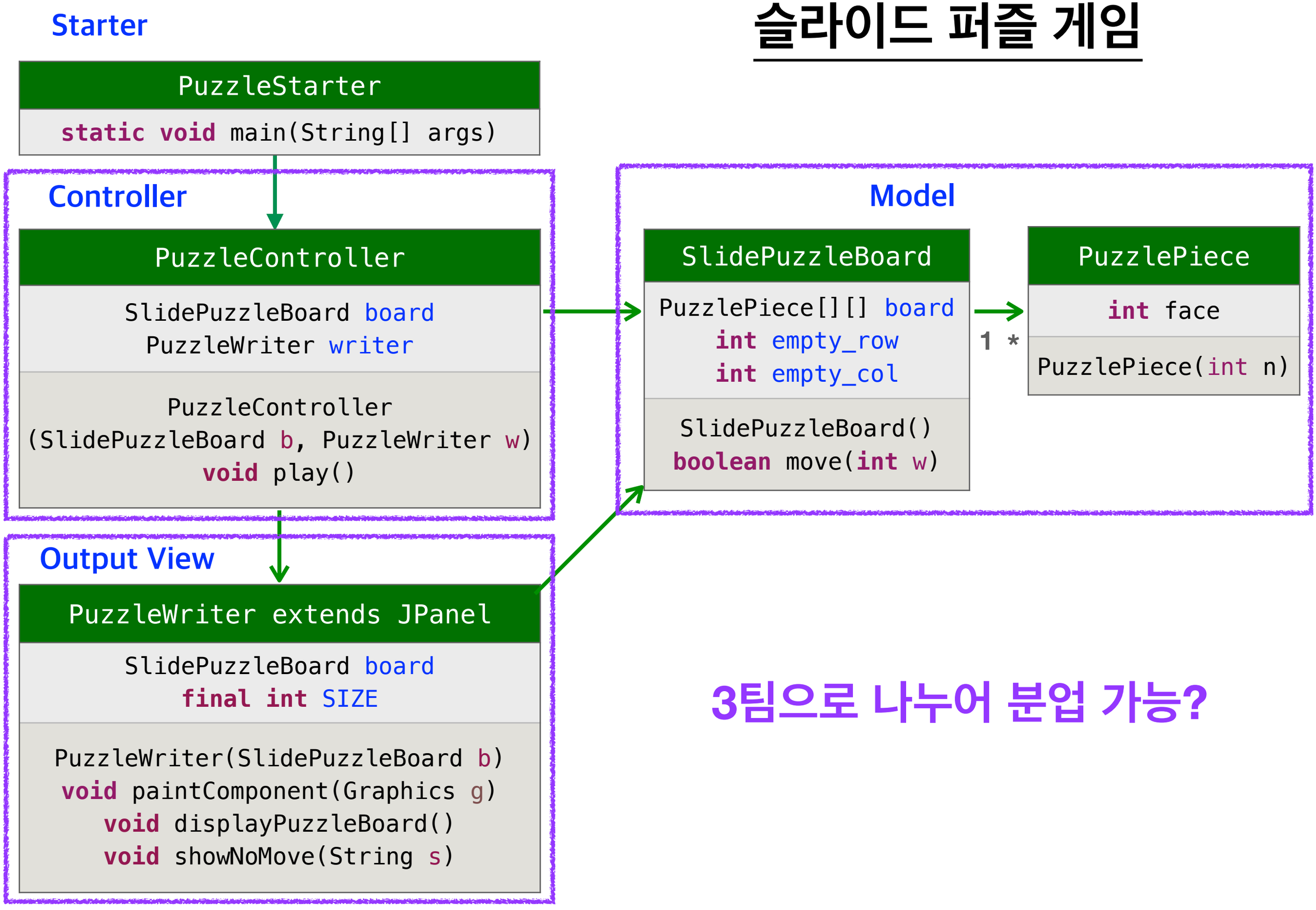
Output View

PuzzleWriter extends JPanel

`SlidePuzzleBoard board`
`final int SIZE`

`PuzzleWriter(SlidePuzzleBoard b)`
`void paintComponent(Graphics g)`
`void displayPuzzleBoard()`
`void showNoMove(String s)`

슬라이드 퍼즐 게임



3팀으로 나누어 분업 가능?

슬라이드 퍼즐 게임

Starter

PuzzleStarter

`static void main(String[] args)`

Controller

PuzzleController

`SlidePuzzleBoard board`
`PuzzleWriter writer`

`PuzzleController (SlidePuzzleBoard b, PuzzleWriter w)`
`void play()`

Output View

PuzzleWriter extends JPanel

`SlidePuzzleBoard board`
`final int SIZE`

`PuzzleWriter(SlidePuzzleBoard b)`
`void paintComponent(Graphics g)`
`void displayPuzzleBoard()`
`void showNoMove(String s)`

Model

SlidePuzzleBoard

`PuzzlePiece[][] board`
`int empty_row`
`int empty_col`

`SlidePuzzleBoard()`
`boolean move(int w)`

1 *

PuzzlePiece

`int face`

`PuzzlePiece(int n)`

클래스 끼리 종속 관계가 있어
구현의 분업이 어려움!

specification

Programming to the interface,

not the implementation

class

프로그래밍 작업 분업 가능

종속 클래스 대신 인터페이스만 있어도 컴파일 가능

슬라이드 퍼즐 게임

Starter

PuzzleStarter

`static void main(String[] args)`

Controller

PuzzleController

`SlidePuzzleBoard board`
`PuzzleWriter writer`

`PuzzleController(SlidePuzzleBoard b, PuzzleWriter w)`
`void play()`

Output View

PuzzleWriter extends JPanel

`SlidePuzzleBoard board`
`final int SIZE`

`PuzzleWriter(SlidePuzzleBoard b)`
`void paintComponent(Graphics g)`
`void displayPuzzleBoard()`
`void showNoMove(String s)`

Model

SlidePuzzleBoard

`PuzzlePiece[][] board`
`int empty_row`
`int empty_col`

`SlidePuzzleBoard()`
`boolean move(int w)`

1 *

PuzzlePiece

`int face`

`PuzzlePiece(int n)`

슬라이드 퍼즐 게임

Starter

PuzzleStarter

`static void main(String[] args)`

Controller

PuzzleController

`SlidePuzzleBoard board`
`PuzzleWriter writer`

`PuzzleController(SlidePuzzleBoard b, PuzzleWriter w)`
`void play()`

Output View

PuzzleWriter extends JPanel

`SlidePuzzleBoard board`
`final int SIZE`

`PuzzleWriter(SlidePuzzleBoard b)`
`void paintComponent(Graphics g)`
`void displayPuzzleBoard()`
`void showNoMove(String s)`

Model

SlidePuzzleBoard

`PuzzlePiece[][] board`
`int empty_row`
`int empty_col`

`SlidePuzzleBoard()`
`boolean move(int w)`

1 *

PuzzlePiece

`int face`

`PuzzlePiece(int n)`

SlidePuzzleBoardInterface

`boolean move(int w)`
`PuzzlePiece[][] board();`

```java
public interface SlidePuzzleBoardInterface {

    public boolean move(int w);

    public PuzzlePiece[][] board();

}
```

슬라이드 퍼즐 게임

Starter

PuzzleStarter

`static void main(String[] args)`

Controller

PuzzleController

`SlidePuzzleBoard board`
`PuzzleWriter writer`

`PuzzleController(SlidePuzzleBoard b, PuzzleWriter w)`
`void play()`

Output View

PuzzleWriter extends JPanel

`SlidePuzzleBoard board`
`final int SIZE`

`PuzzleWriter(SlidePuzzleBoard b)`
`void paintComponent(Graphics g)`
`void displayPuzzleBoard()`
`void showNoMove(String s)`

Model

SlidePuzzleBoard

`PuzzlePiece[][] board`
`int empty_row`
`int empty_col`

`SlidePuzzleBoard()`
`boolean move(int w)`

1 *

PuzzlePiece

`int face`

`PuzzlePiece(int n)`

implements

SlidePuzzleBoardInterface

`boolean move(int w)`
`PuzzlePiece[][] board();`

```java
public interface SlidePuzzleBoardInterface {

    public boolean move(int w);

    public PuzzlePiece[][] board();

}
```

```java
public class SlidePuzzleBoard implements SlidePuzzleBoardInterface {

    private PuzzlePiece[][] board;
    private int empty_row;
    private int empty_col;

    public PuzzlePiece[][] board() { return board; }

    public SlidePuzzleBoard() {
        board = new PuzzlePiece[4][4];
        int number = 15;
        for (int i = 0; i < 4; i++)
            for (int j = 0; j < 4; j++) {
                board[i][j] = new PuzzlePiece(number);
                number -= 1;
            }
        board[3][3] = null;
        empty_row = 3;
        empty_col = 3;
    }

    public boolean move(int n) {
        int row, col;
        if (found(n,empty_row-1, empty_col)) {
            row = empty_row-1;
```

슬라이드 퍼즐 게임

Starter

PuzzleStarter

```
static void main(String[] args)
```

Controller

PuzzleController

```
SlidePuzzleBoard board
PuzzleWriter writer
PuzzleController
(SlidePuzzleBoard b, PuzzleWriter w)
void play()
```

Output View

PuzzleWriter extends JPanel

```
SlidePuzzleBoard board
final int SIZE
PuzzleWriter(SlidePuzzleBoard b)
void paintComponent(Graphics g)
void displayPuzzleBoard()
void showNoMove(String s)
```

Model

SlidePuzzleBoard

```
PuzzlePiece[][] board
int empty_row
int empty_col
SlidePuzzleBoard()
boolean move(int w)
```

1 *

PuzzlePiece

```
int face
PuzzlePiece(int n)
```

implements

SlidePuzzleBoardInterface

```
boolean move(int w)
PuzzlePiece[][] board();
```

```java
public interface SlidePuzzleBoardInterface {

    public boolean move(int w);

    public PuzzlePiece[][] board();

}
```

```java
import javax.swing.*;

public class PuzzleController {

    private SlidePuzzleBoard puzzle;
    private PuzzleWriterInterface writer;

    public PuzzleController(SlidePuzzleBoard p, PuzzleWriter w) {
        puzzle = p;
        writer = w;
    }

    public void play() {
        while (true) {
            writer.displayPuzzleBoard();
            String input = JOptionPane.showInputDialog("Your move:");
            int n = Integer.parseInt(input);
            if (! puzzle.move(n))
                writer.showNoMove("Cannot move.");
        }
    }

}
```

SlidePuzzleBoardInterface

```java
public interface SlidePuzzleBoardInterface {

    public boolean move(int w);

    public PuzzlePiece[][] board();

}
```

```java
import java.awt.*;
import javax.swing.*;

public class PuzzleWriter extends JPanel {
    private SlidePuzzleBoard puzzle;
    private final int SIZE; // the size of a puzzle piece in pixel

    public PuzzleWriter(SlidePuzzleBoard p) {
        puzzle = p;
        SIZE = 30;
        JFrame f = new JFrame();
        f.getContentPane().add(this);
        f.setLocation(550,100);
        f.setTitle("Slide Puzzle");
        f.setSize(SIZE*6, SIZE*6+28);
        f.setVisible(true);
        f.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }

    public void paintComponent(Graphics g) {
        g.setColor(Color.YELLOW);
        g.fillRect(0, 0, SIZE*6, SIZE*6);
        PuzzlePiece[][] r = puzzle.board();
        for (int i = 0; i < 4; i = i + 1) {
            for (int j = 0; j < 4; j = j + 1) {
```

SlidePuzzleBoardInterface

슬라이드 퍼즐 게임

Starter

PuzzleStarter

```
static void main(String[] args)
```

Controller

PuzzleController

```
SlidePuzzleBoardInterface board
PuzzleWriter writer
PuzzleController
(SlidePuzzleBoardInterface b,
PuzzleWriter w)
void play()
```

Model

SlidePuzzleBoard

```
PuzzlePiece[][] board
int empty_row
int empty_col
SlidePuzzleBoard()
boolean move(int w)
```

PuzzlePiece

```
int face
PuzzlePiece(int n)
```

1 *

implements

SlidePuzzleBoardInterface

```
boolean move(int w)
PuzzlePiece[][] board();
```

Output View

PuzzleWriter extends JPanel

```
SlidePuzzleBoardInterface board
final int SIZE
PuzzleWriter(SlidePuzzleBoardInterface b)
void paintComponent(Graphics g)
void displayPuzzleBoard()
void showNoMove(String s)
```

슬라이드 퍼즐 게임

Starter

PuzzleStarter

```
static void main(String[] args)
```

Controller

PuzzleController

```
SlidePuzzleBoardInterface board
PuzzleWriter writer
PuzzleController
(SlidePuzzleBoardInterface b,
PuzzleWriter w)
void play()
```

Output View

PuzzleWriter extends JPanel

```
SlidePuzzleBoardInterface board
final int SIZE
PuzzleWriter(SlidePuzzleBoardInterface b)
void paintComponent(Graphics g)
void displayPuzzleBoard()
void showNoMove(String s)
```

Model

SlidePuzzleBoard

```
PuzzlePiece[][] board
int empty_row
int empty_col
SlidePuzzleBoard()
boolean move(int w)
```

1 *

PuzzlePiece

```
int face
PuzzlePiece(int n)
```

implements

SlidePuzzleBoardInterface

```
boolean move(int w)
PuzzlePiece[][] board();
```

슬라이드 퍼즐 게임

Starter

PuzzleStarter

```
static void main(String[] args)
```

Controller

PuzzleController

```
SlidePuzzleBoardInterface board
PuzzleWriter writer
PuzzleController
(SlidePuzzleBoardInterface b,
PuzzleWriter w)
void play()
```

Output View

PuzzleWriter extends JPanel

```
SlidePuzzleBoardInterface board
final int SIZE
PuzzleWriter(SlidePuzzleBoardInterface b)
void paintComponent(Graphics g)
void displayPuzzleBoard()
void showNoMove(String s)
```

Model

SlidePuzzleBoard

```
PuzzlePiece[][] board
int empty_row
int empty_col
SlidePuzzleBoard()
boolean move(int w)
```

1 *

PuzzlePiece

```
int face
PuzzlePiece(int n)
```

implements

SlidePuzzleBoardInterface

```
boolean move(int w)
PuzzlePiece[][] board();
```

PuzzleWriterInterface

```
void displayPuzzleBoard()
void showNoMove(String s)
```

```java
public interface PuzzleWriterInterface {

    public void displayPuzzleBoard();

    public void showNoMove(String s);

}
```

슬라이드 퍼즐 게임

Starter

PuzzleStarter

```
static void main(String[] args)
```

Controller

PuzzleController

```
SlidePuzzleBoardInterface board
PuzzleWriter writer
PuzzleController
(SlidePuzzleBoardInterface b,
PuzzleWriter w)
void play()
```

Output View

PuzzleWriter extends JPanel

```
SlidePuzzleBoardInterface board
final int SIZE
PuzzleWriter(SlidePuzzleBoardInterface b)
void paintComponent(Graphics g)
void displayPuzzleBoard()
void showNoMove(String s)
```

Model

SlidePuzzleBoard

```
PuzzlePiece[][] board
int empty_row
int empty_col
SlidePuzzleBoard()
boolean move(int w)
```

1 *

PuzzlePiece

```
int face
PuzzlePiece(int n)
```

implements

SlidePuzzleBoardInterface

```
boolean move(int w)
PuzzlePiece[][] board();
```

implements

PuzzleWriterInterface

```
void displayPuzzleBoard()
void showNoMove(String s)
```

```java
public interface PuzzleWriterInterface {

    public void displayPuzzleBoard();

    public void showNoMove(String s);

}
```

```java
import java.awt.*;
import javax.swing.*;

public class PuzzleWriter extends JPanel implements PuzzleWriterInterface {
    private SlidePuzzleBoardInterface puzzle;
    private final int SIZE; // the size of a puzzle piece in pixel

    public PuzzleWriter(SlidePuzzleBoardInterface p) {
        puzzle = p;
        SIZE = 30;
        JFrame f = new JFrame();
        f.getContentPane().add(this);
        f.setLocation(550,100);
        f.setTitle("Slide Puzzle");
        f.setSize(SIZE*6, SIZE*6+28);
        f.setVisible(true);
        f.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }

    public void paintComponent(Graphics g) {
        g.setColor(Color.YELLOW);
        g.fillRect(0, 0, SIZE*6, SIZE*6);
        PuzzlePiece[][] r = puzzle.board();
        for (int i = 0; i < 4; i = i + 1) {
            for (int j = 0; j < 4; j = j + 1) {
                paintPiece(g, r[i][j], i, j);
```

슬라이드 퍼즐 게임

Starter

PuzzleStarter

```
static void main(String[] args)
```

Controller

PuzzleController

```
SlidePuzzleBoardInterface board
PuzzleWriter writer
PuzzleController
(SlidePuzzleBoardInterface b,
PuzzleWriter w)
void play()
```

Output View

PuzzleWriter extends JPanel

```
SlidePuzzleBoardInterface board
final int SIZE
PuzzleWriter(SlidePuzzleBoardInterface b)
void paintComponent(Graphics g)
void displayPuzzleBoard()
void showNoMove(String s)
```

Model

SlidePuzzleBoard

```
PuzzlePiece[][] board
int empty_row
int empty_col
SlidePuzzleBoard()
boolean move(int w)
```

1 *

PuzzlePiece

```
int face
PuzzlePiece(int n)
```

implements

SlidePuzzleBoardInterface

```
boolean move(int w)
PuzzlePiece[][] board();
```

PuzzleWriterInterface

```
void displayPuzzleBoard()
void showNoMove(String s)
```

implements

```java
public interface PuzzleWriterInterface {

    public void displayPuzzleBoard();

    public void showNoMove(String s);

}
```

```java
import javax.swing.*;

public class PuzzleController {

    private SlidePuzzleBoardInterface puzzle;
    private PuzzleWriter writer;

    public PuzzleController(SlidePuzzleBoardInterface p, PuzzleWriter w) {
        puzzle = p;
        writer = w;
    }

    public void play() {
        while (true) {
            writer.displayPuzzleBoard();
            String input = JOptionPane.showInputDialog("Your move:");
            int n = Integer.parseInt(input);
            if (! puzzle.move(n))
                writer.showNoMove("Cannot move.");
        }
    }

}
```

PuzzleWriterInterface

슬라이드 퍼즐 게임

Starter

PuzzleStarter

```
static void main(String[] args)
```

Controller

PuzzleController

```
SlidePuzzleBoardInterface board
PuzzleWriterInterface writer
PuzzleController
(SlidePuzzleBoardInterface b,
PuzzleWriterInterface w)
void play()
```

Model

SlidePuzzleBoard

```
PuzzlePiece[][] board
int empty_row
int empty_col
SlidePuzzleBoard()
boolean move(int w)
```

1 *

PuzzlePiece

```
int face
PuzzlePiece(int n)
```

implements

Output View

PuzzleWriter extends JPanel

```
SlidePuzzleBoardInterface board
final int SIZE
PuzzleWriter(SlidePuzzleBoardInterface b)
void paintComponent(Graphics g)
void displayPuzzleBoard()
void showNoMove(String s)
```

implements

SlidePuzzleBoardInterface

```
boolean move(int w)
PuzzlePiece[][] board();
```

PuzzleWriterInterface

```
void displayPuzzleBoard()
void showNoMove(String s)
```

상속 Inheritance	구현 Implementation
`class Sub extends Super`	`class C implements S`
• Sub = 하위 클래스 • Super = 상위 클래스 • 클래스 Super를 상속 받아 클래스 Sub를 구현 • 클래스 Super의 코드를 클래스 Sub에서 재사용	• 명세(interface) S를 만족하는 클래스 C 구현 • `S x = new C();`

사례 학습 : 상속 Inheritance 이해하기

```
public class Person {

    private String name;

    public Person(String n) {
        name = n;
    }

    public String name() {
        return name;
    }

    . . . . . .
}
```

```
public class PersonFrom extends Person {

    private String city;

    public PersonFrom(String n, String c) {
        super(n);
        city = c;
    }

    public String city() {
        return city;
    }

    . . . . . .
}
```

```
PersonFrom x = new PersonFrom("마음", “서울");
System.out.println("이름: " + x.name());
System.out.println("출신: " + x.city());
```

```java
public class Person {
    private String name;

    public Person(String n) {
        name = n;
    }

    public String name() {
        return name;
    }

    public boolean sameName(Person other) {
        return name.equals(other.name());
    }
}
```

```java
public class PersonFrom extends Person {
    private String city;

    public PersonFrom(String n, String c) {
        super(n);
        city = c;
    }

    public String city() {
        return city;
    }

    public boolean same(PersonFrom other) {
        return sameName(other) &&
               city.equals(other.city());
    }
}
```

```java
Person p = new Person("마음");
Person q = new PersonFrom("소리","서울");
```

다음 중에서 어떤 문장이 Java 컴파일러를 통과할까?
통과하여 실행하면, 무엇을 프린트할까?

- `System.out.println(p.city());`
- `System.out.println(q.name());`
- `System.out.println(p.sameName(p));`
- `System.out.println(q.sameName(p));`
- `System.out.println(q.same(p));`

Subtyping

기본 타입

byte **int** **long** **float** **double** **boolean**

byte <= **int** <= **long** <= **float** <= **double**
int는 **double**의 서브타입 이다. **double** 값을 담을 수 있는 변수에 **int** 값도 담을 수 있다.

예:

```
double d = 3;
```

```
public double inverseOf(double d) {
    return 1.0 / d
}
System.out.println(inverseOf(3));
```

Subtyping

객체 타입

Object Types

Reference Types

```
public class MyPanel extends JPanel
```

MyPanel <= JPanel

MyPanel은 **JPanel**의 서브타입 이다.

JPanel 객체을 담을 수 있는 변수에 **MyPanel** 객체도 담을 수 있다.

상속

inheritance

예:

```
JFrame f = new JFrame();
f.getContentPane().add(new MyPanel());
```

JFrame
Container getContentPane()

Container
Component add(Component comp)

MyPanel <= JPanel <= JComponent <= Container <= Component <= Object

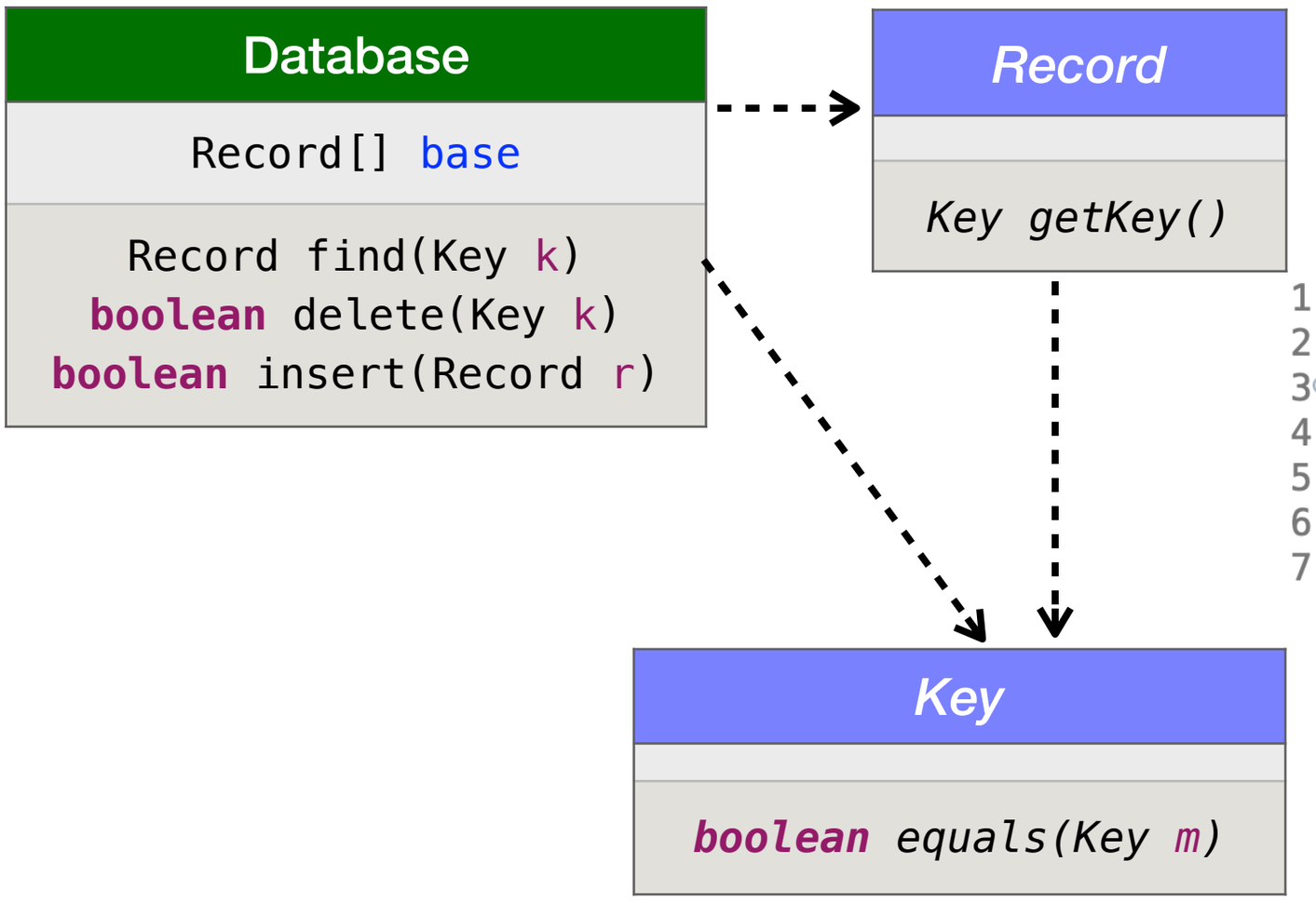
```java
public class PuzzleStarter {

    public static void main(String[] args) {
        SlidePuzzleBoard puzzle = new SlidePuzzleBoard();
        PuzzleWriter writer = new PuzzleWriter(puzzle);
        new PuzzleController(puzzle, writer).play();
    }

}
```

SlidePuzzleBoard <= SlidePuzzleBoardInterface

PuzzleWriter <= PuzzleWriterInterface

```java
import javax.swing.*;

public class PuzzleController {

    private SlidePuzzleBoardInterface puzzle;
    private PuzzleWriterInterface writer;

    public PuzzleController(SlidePuzzleBoardInterface p, PuzzleWriterInterface w) {
        puzzle = p;
        writer = w;
    }

    public void play() {
        while (true) {
            writer.displayPuzzleBoard();
            String input = JOptionPane.showInputDialog("Your move:");
            int n = Integer.parseInt(input);
            if (! puzzle.move(n))
                writer.showNoMove("Cannot move.");
        }
    }

}
```

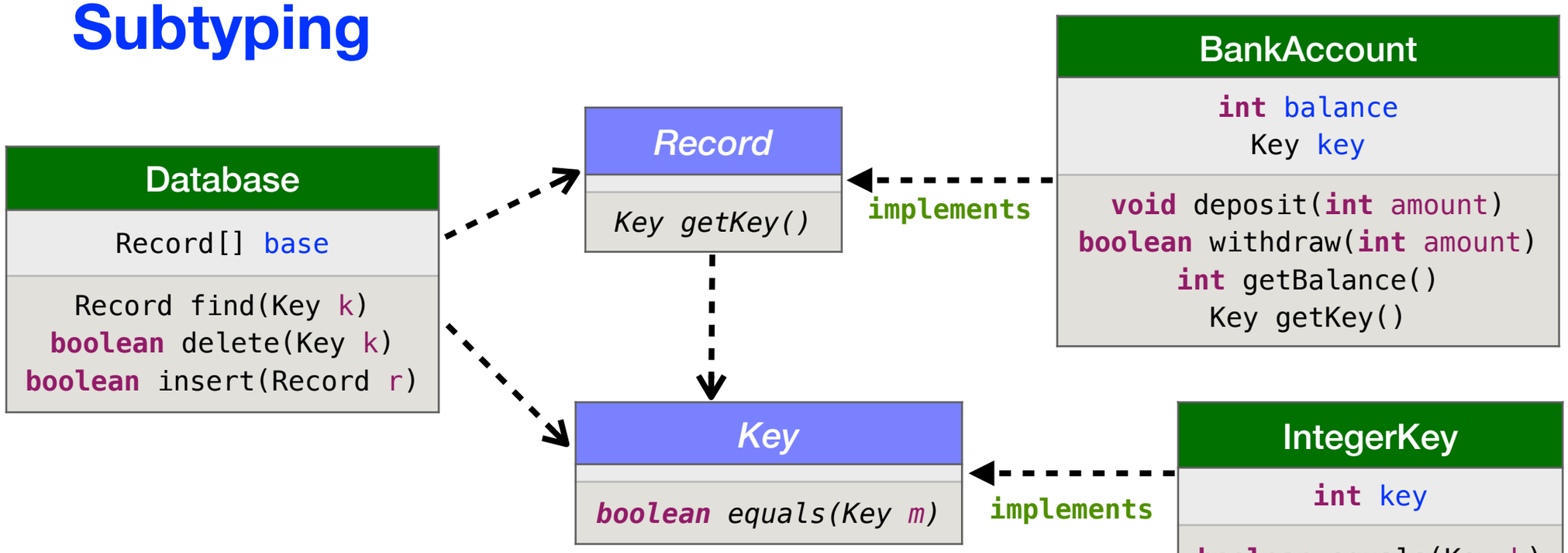
Architecture for a Database




```java
public interface Record {

    /** getKey - 레코드를 유일하게 식별하는 키를 리턴
     * @return - 키 */
    public Key getKey();

}
```

```java
public interface Key {

    /** equals - 인수로 제공된 키와 자신과 같은지 비교
     * @param - 비교 대상 키
     * @return - 같으면 true, 다르면 false */
    public boolean equals(Key m);
}
```

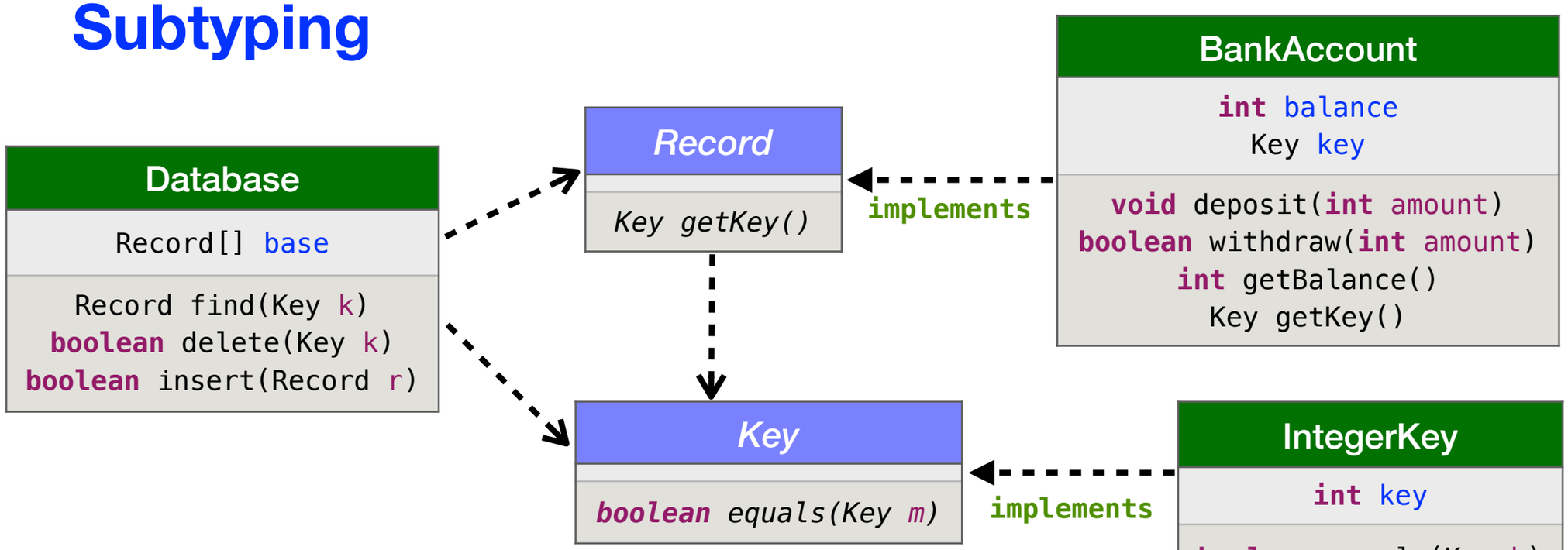
Subtyping




```
public class BankAccount implements Record
```

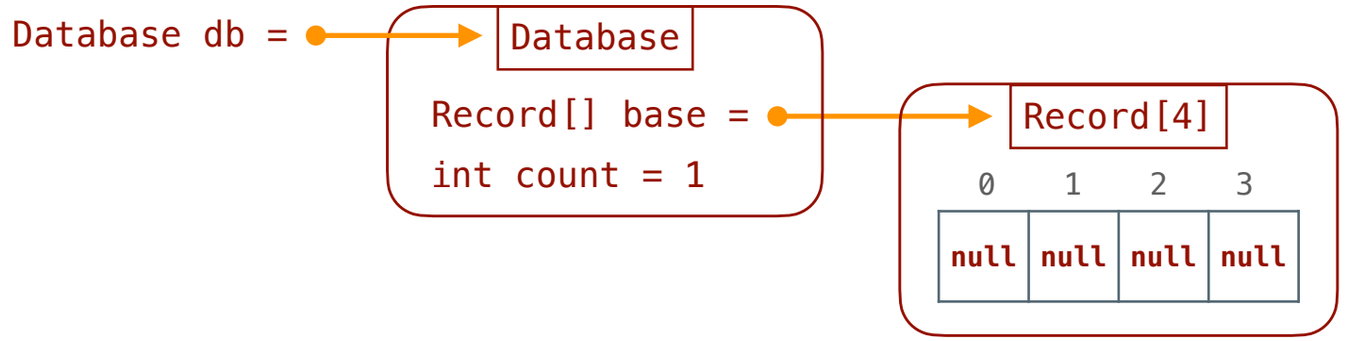
```
Database db = new Database(4);
IntegerKey k = new IntegerKey(55);
BankAccount b = new BankAccount(10000, k);
boolean success = db.insert(b);
```

Subtyping

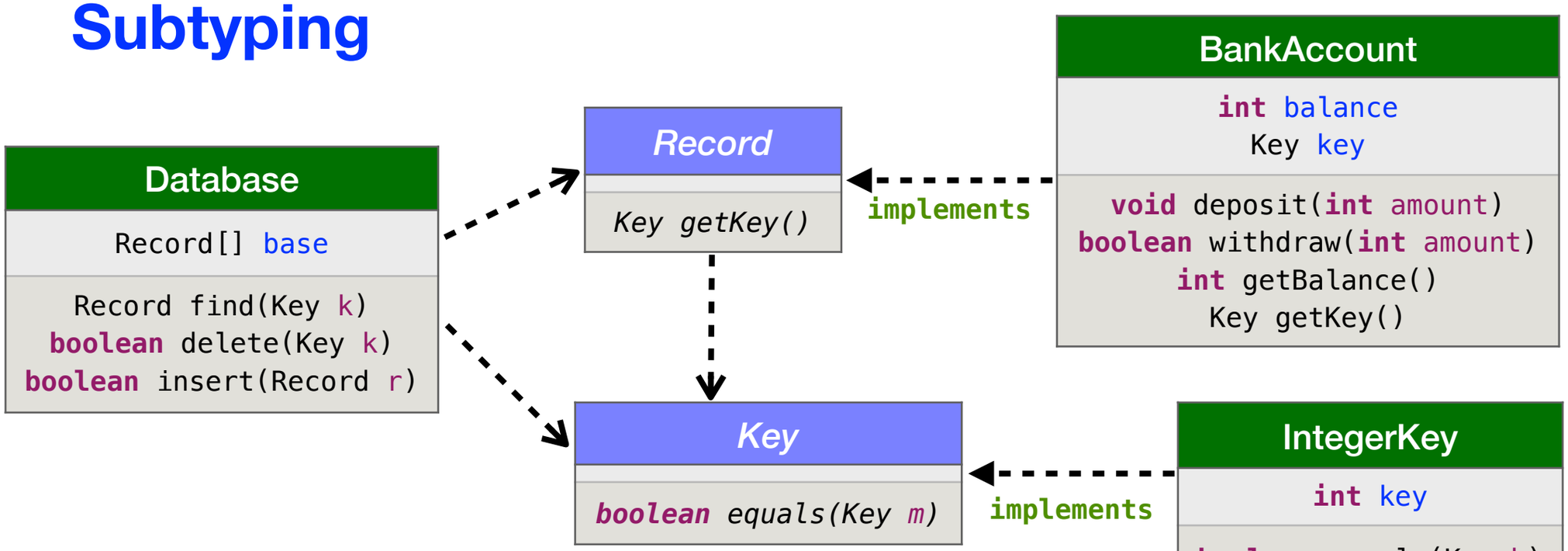



```
public class BankAccount implements Record
```

```
Database db = new Database(4);
IntegerKey k = new IntegerKey(55);
BankAccount b = new BankAccount(10000, k);
boolean success = db.insert(b);
```

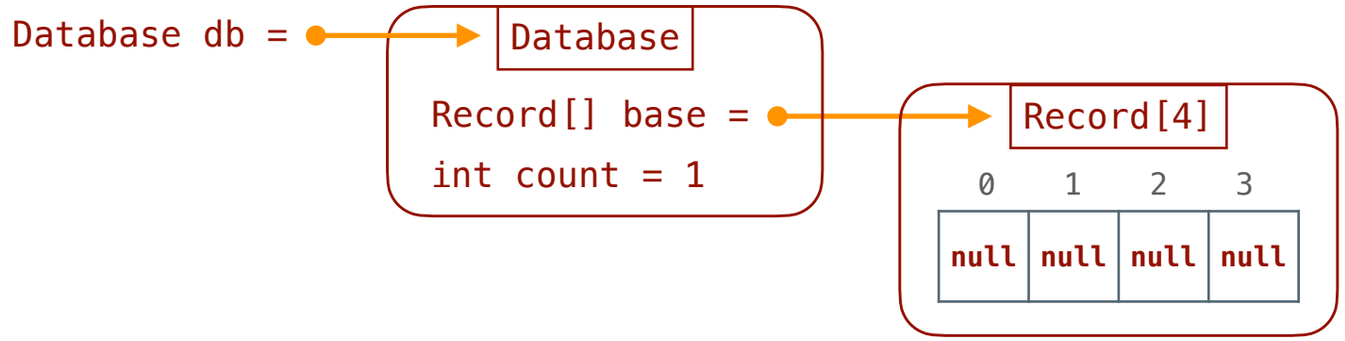


Subtyping

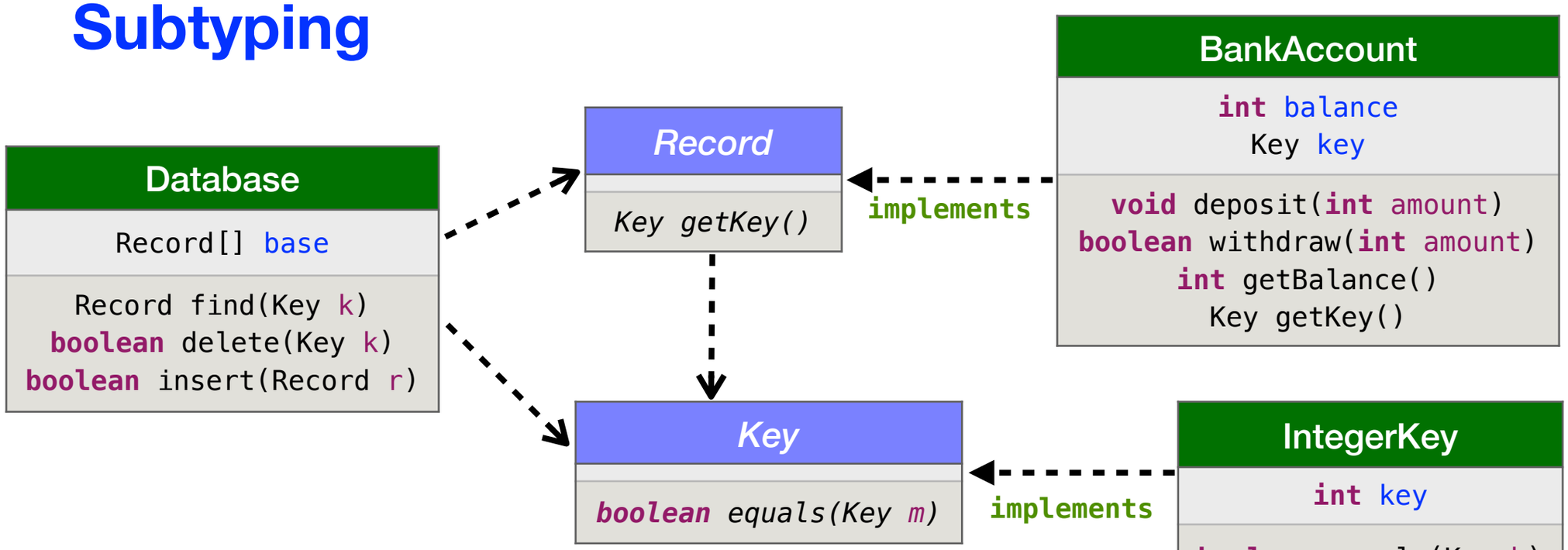



```
public class BankAccount implements Record
```

```
Database db = new Database(4);
IntegerKey k = new IntegerKey(55);
BankAccount b = new BankAccount(10000, k);
boolean success = db.insert(b);
```

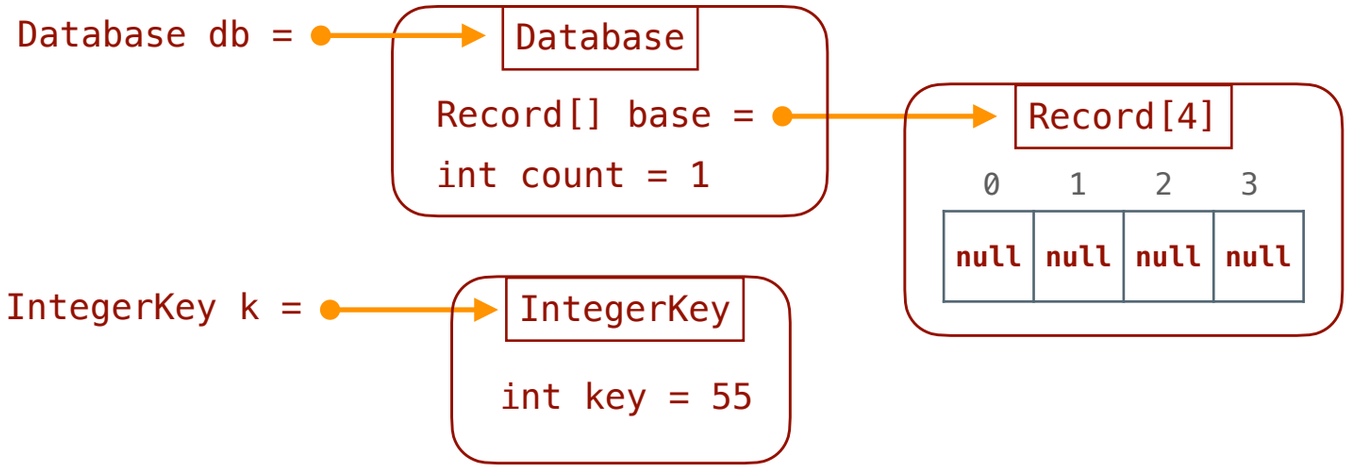


Subtyping

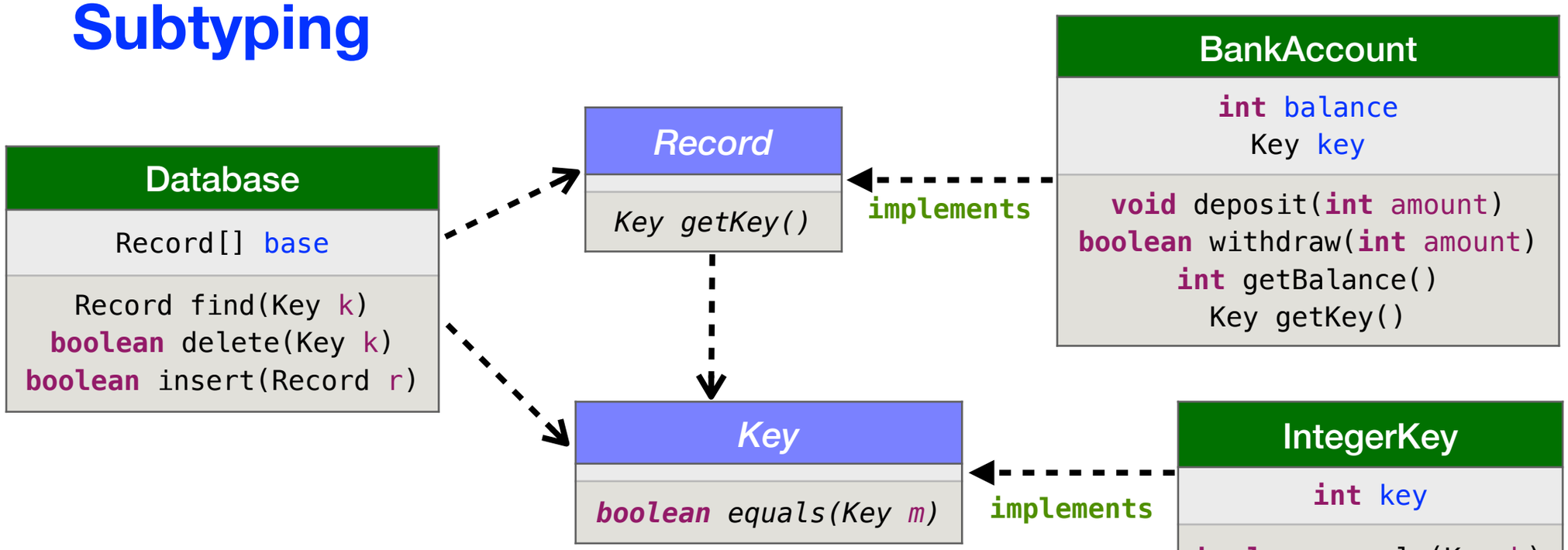



```
public class BankAccount implements Record
```

```
Database db = new Database(4);
IntegerKey k = new IntegerKey(55);
BankAccount b = new BankAccount(10000, k);
boolean success = db.insert(b);
```

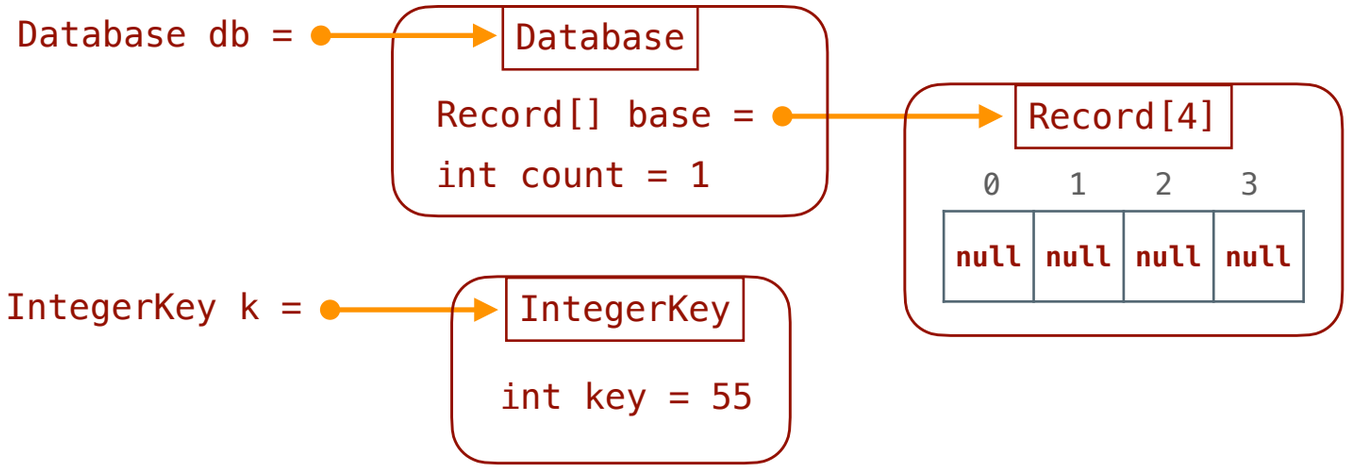


Subtyping

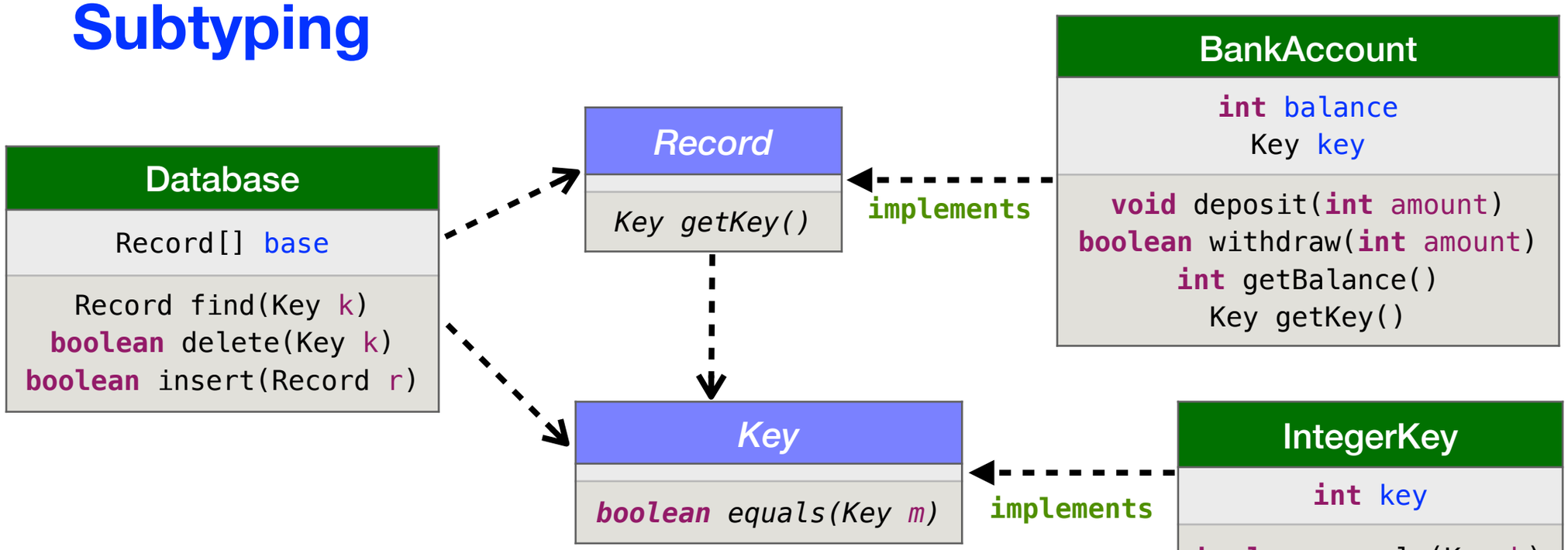



```
public class BankAccount implements Record
```

```
Database db = new Database(4);
IntegerKey k = new IntegerKey(55);
BankAccount b = new BankAccount(10000, k);
boolean success = db.insert(b);
```

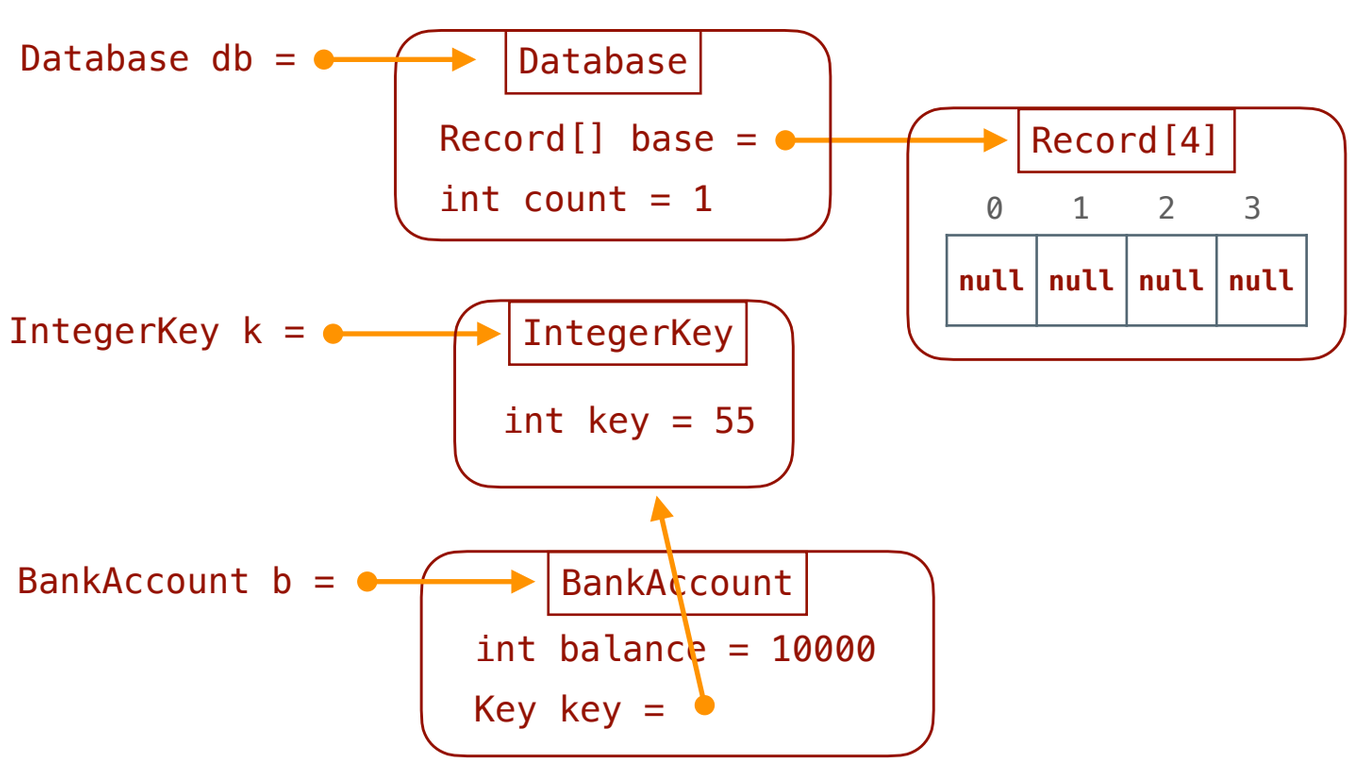


Subtyping

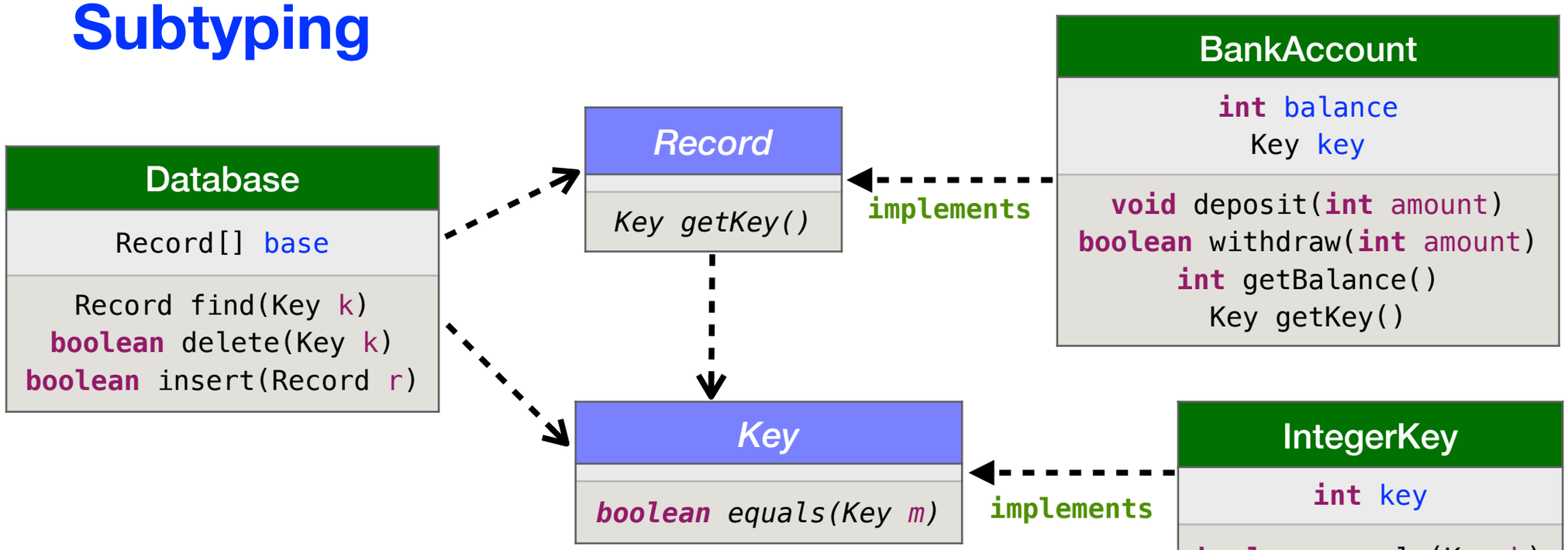



```
public class BankAccount implements Record
```

```
Database db = new Database(4);
IntegerKey k = new IntegerKey(55);
BankAccount b = new BankAccount(10000, k);
boolean success = db.insert(b);
```

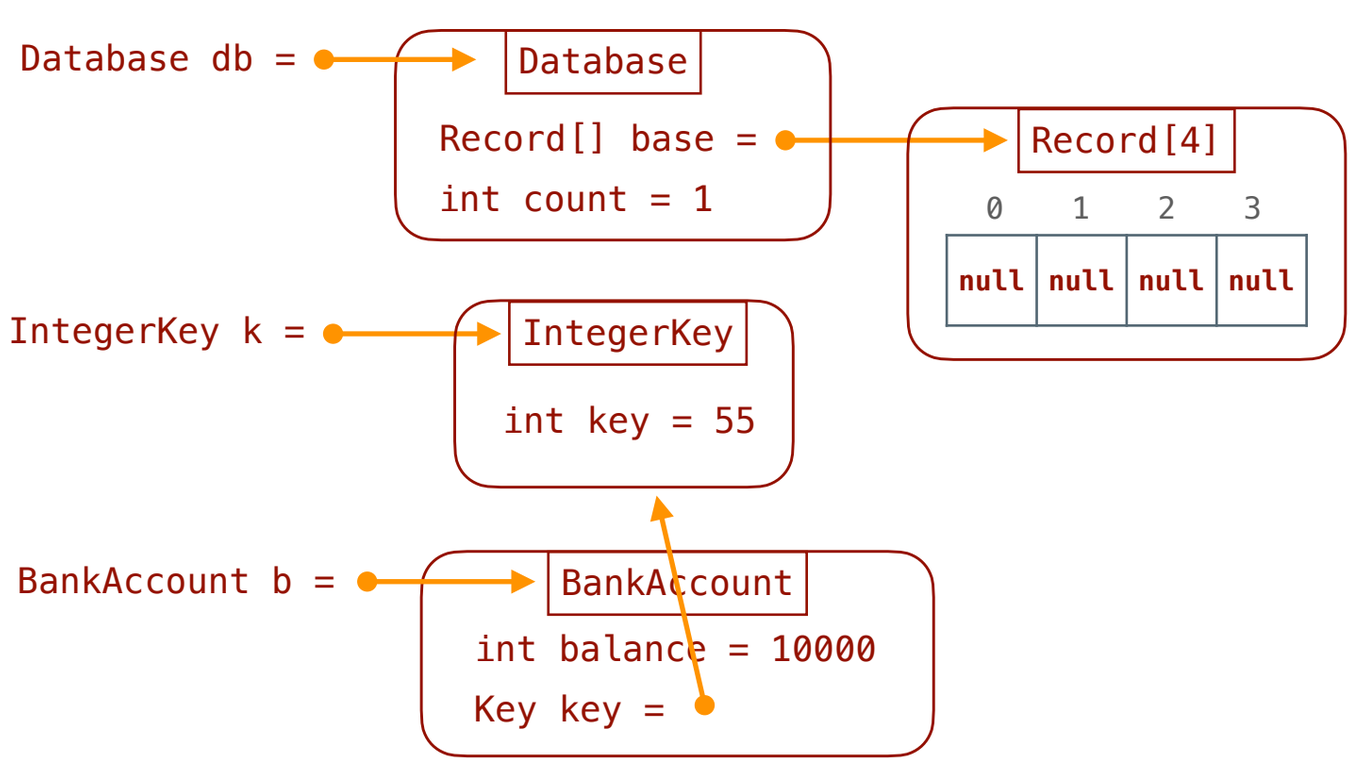


Subtyping

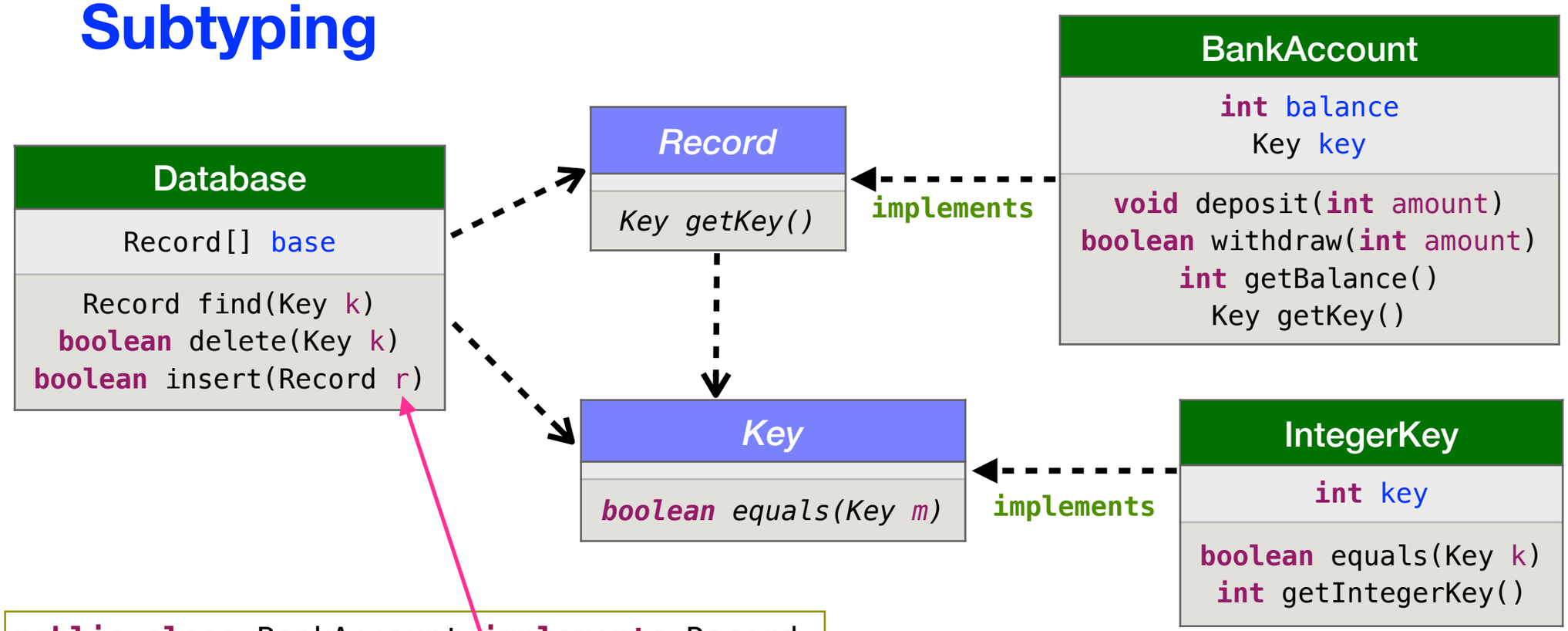



```
public class BankAccount implements Record
```

```
Database db = new Database(4);
IntegerKey k = new IntegerKey(55);
BankAccount b = new BankAccount(10000, k);
boolean success = db.insert(b);
```

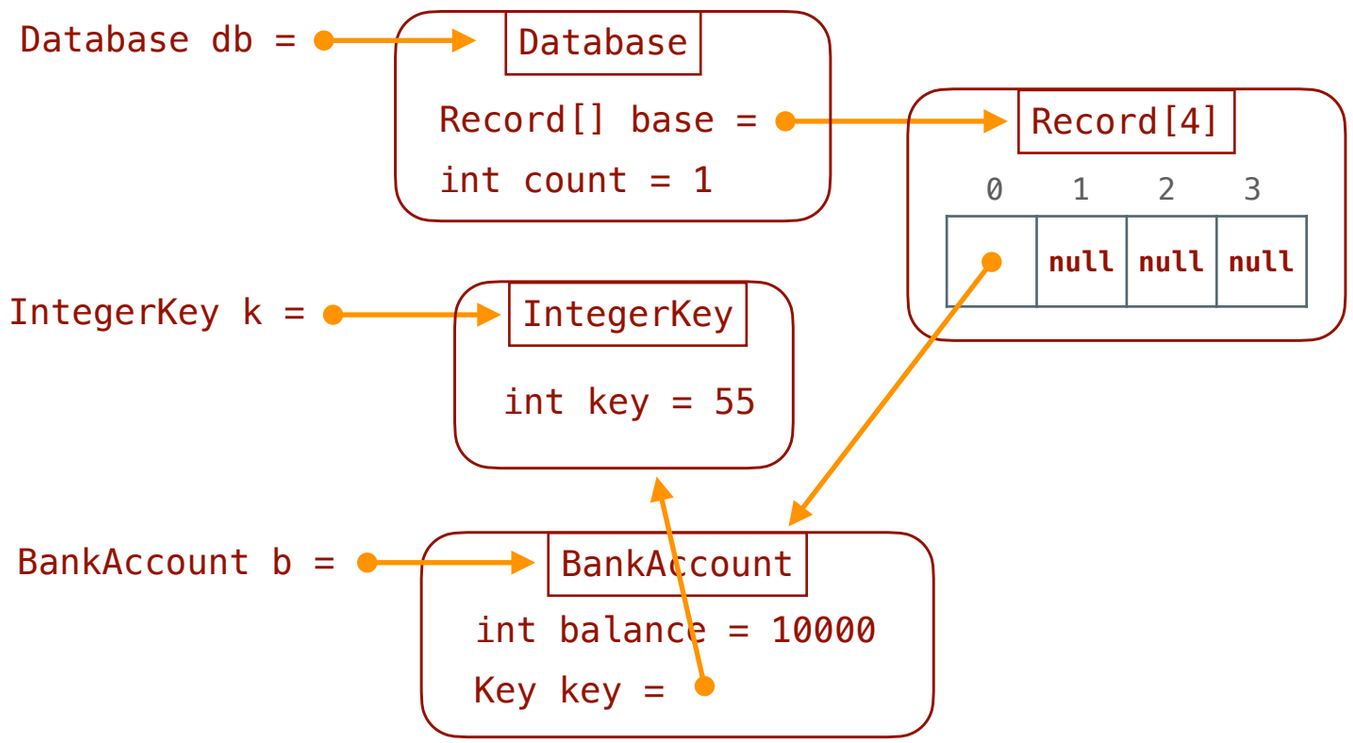


Subtyping

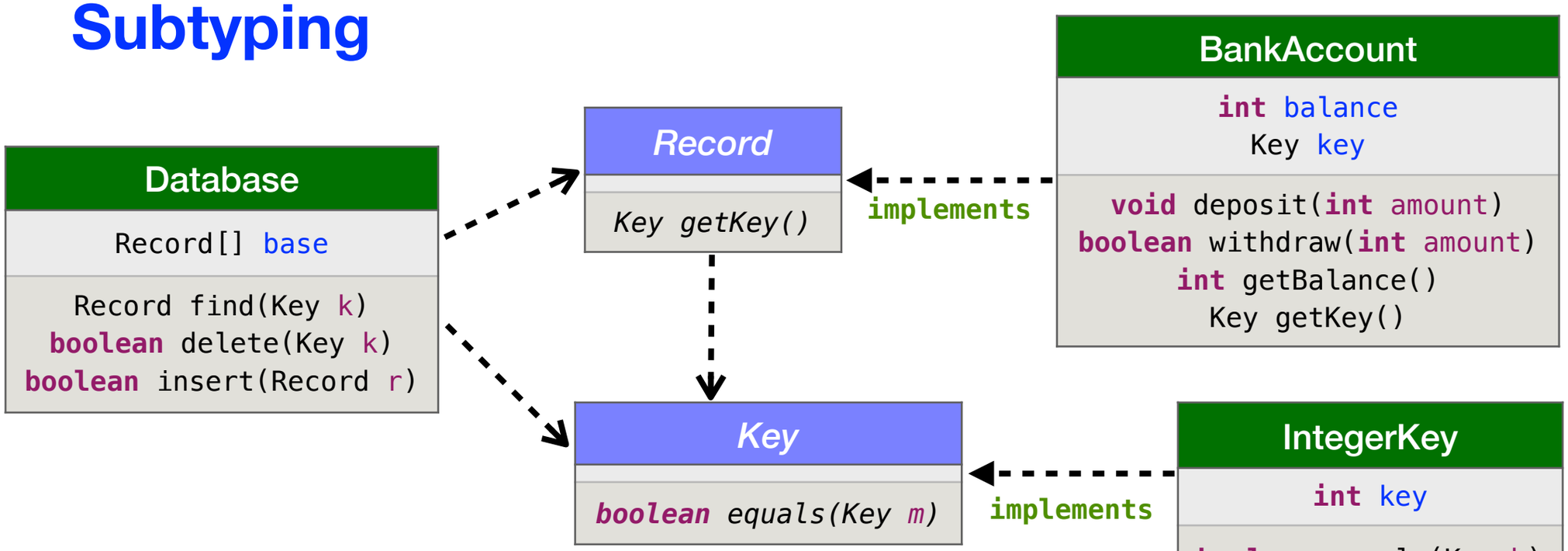



```
public class BankAccount implements Record
```

```
Database db = new Database(4);
IntegerKey k = new IntegerKey(55);
BankAccount b = new BankAccount(10000, k);
boolean success = db.insert(b);
```

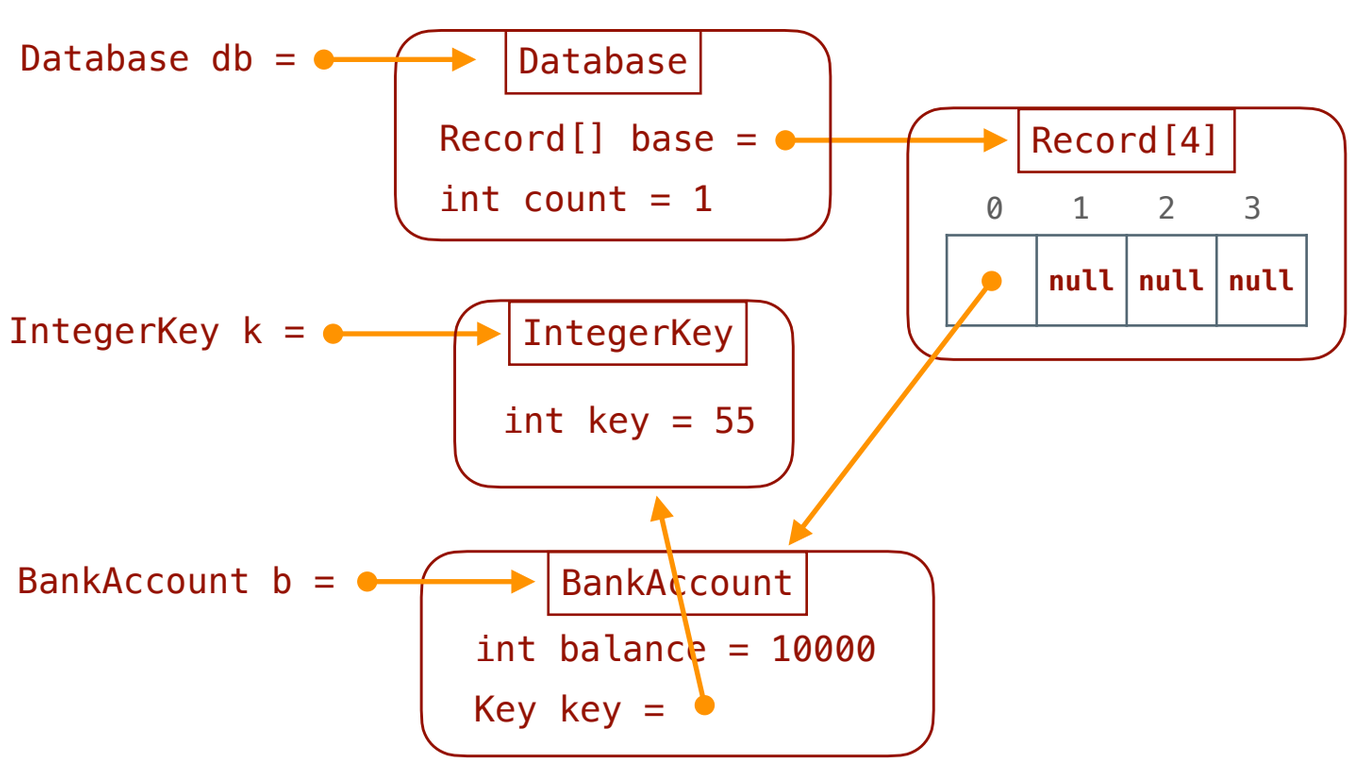


Subtyping

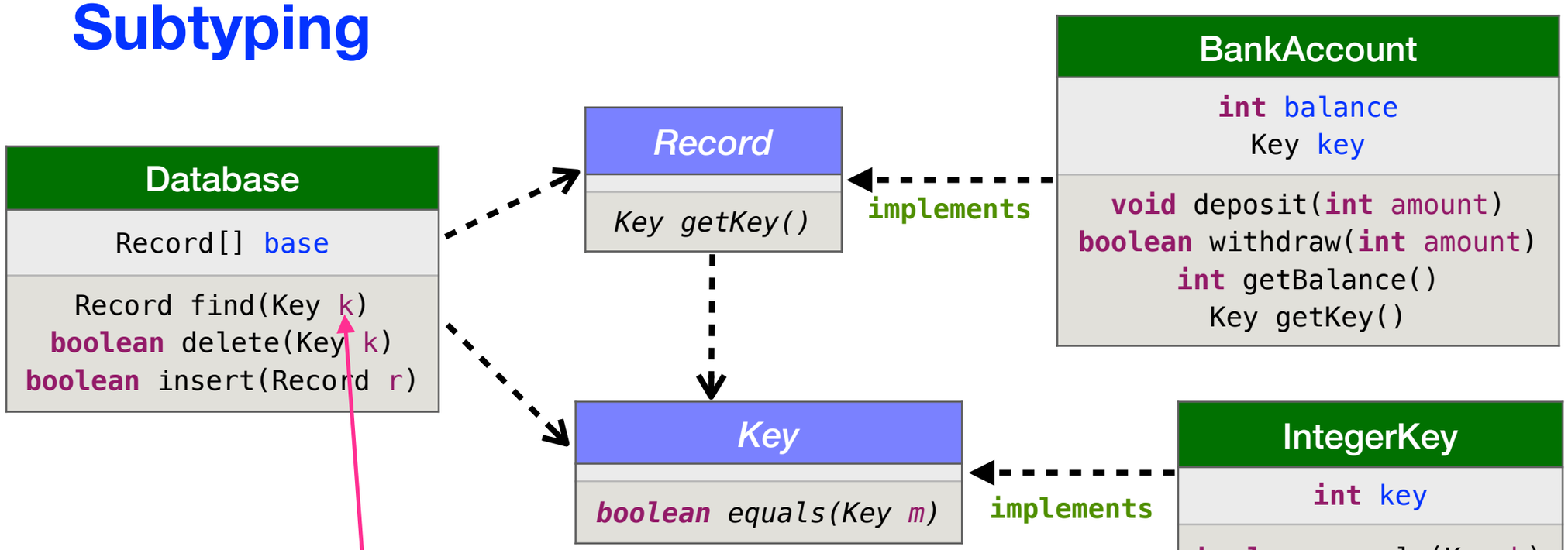



```
public class IntegerKey implements Key
```

```
Record r = db.find(k);

… r.getBalance() …

… ((BankAccount)r).getBalance() …;
```

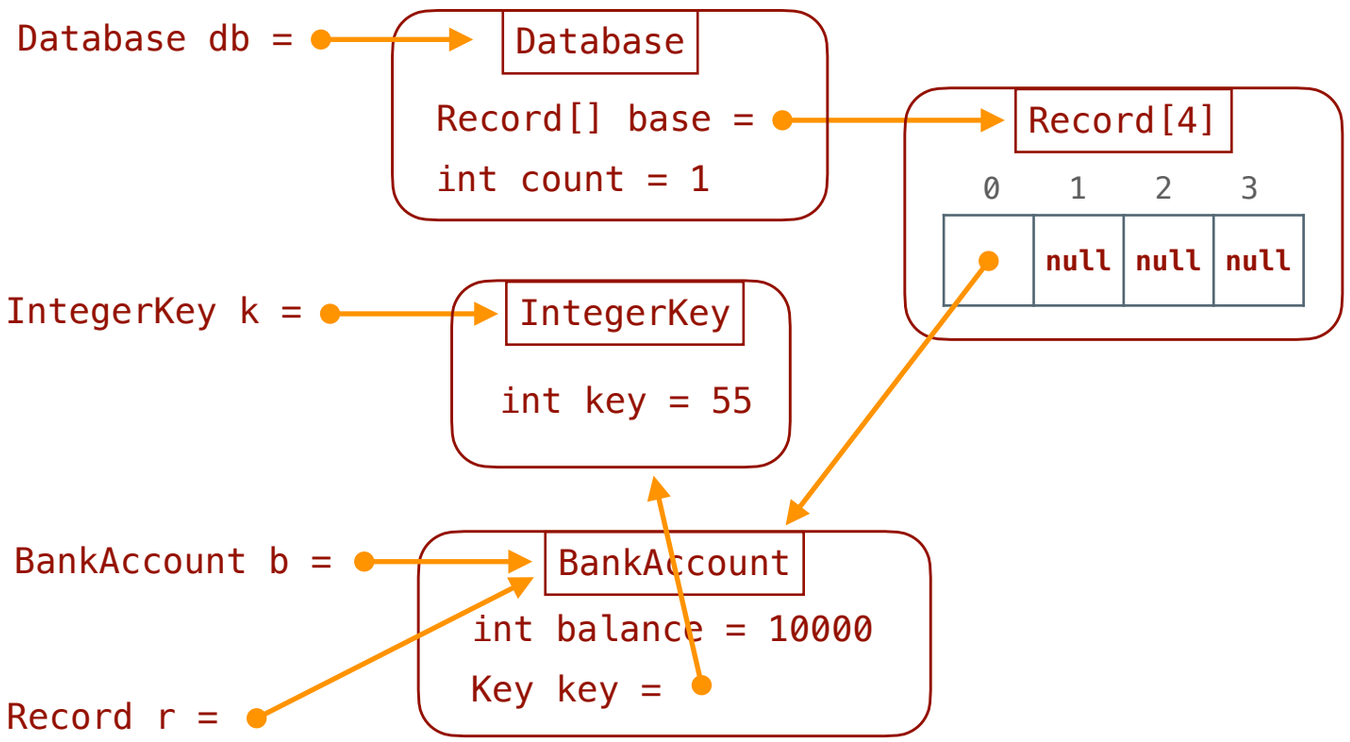


Subtyping

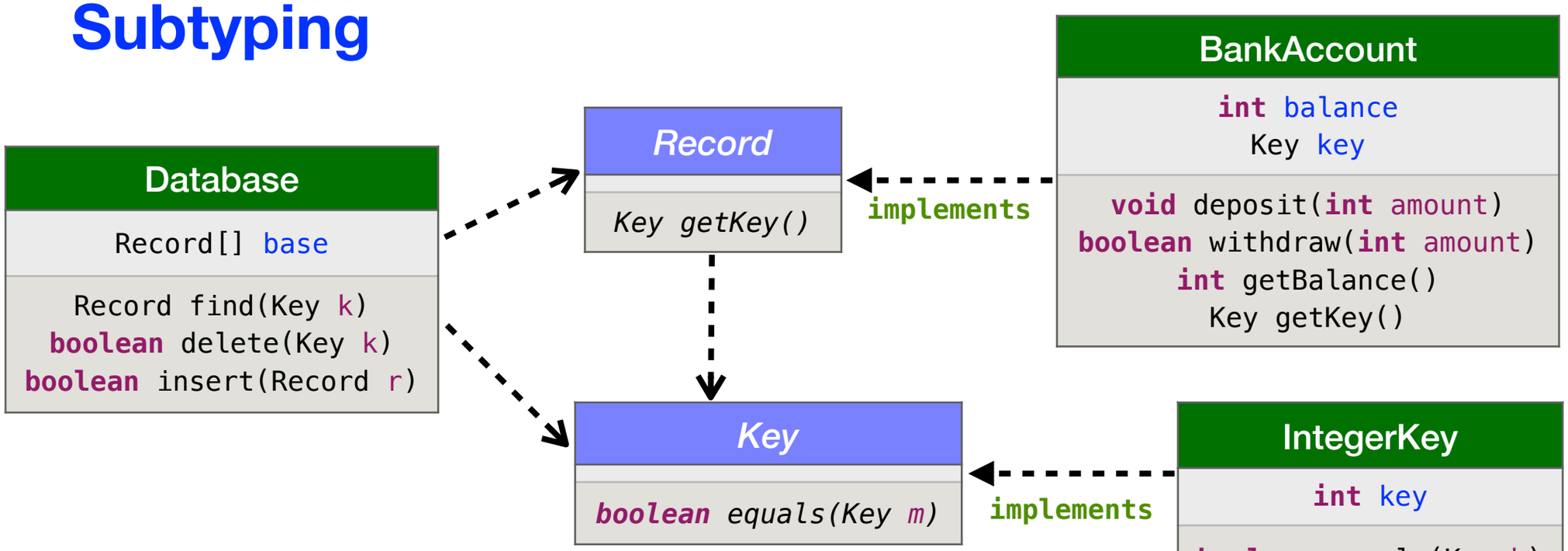



```
public class IntegerKey implements Key
```

```
Record r = db.find(k);

… r.getBalance() …

… ((BankAccount)r).getBalance() …;
```

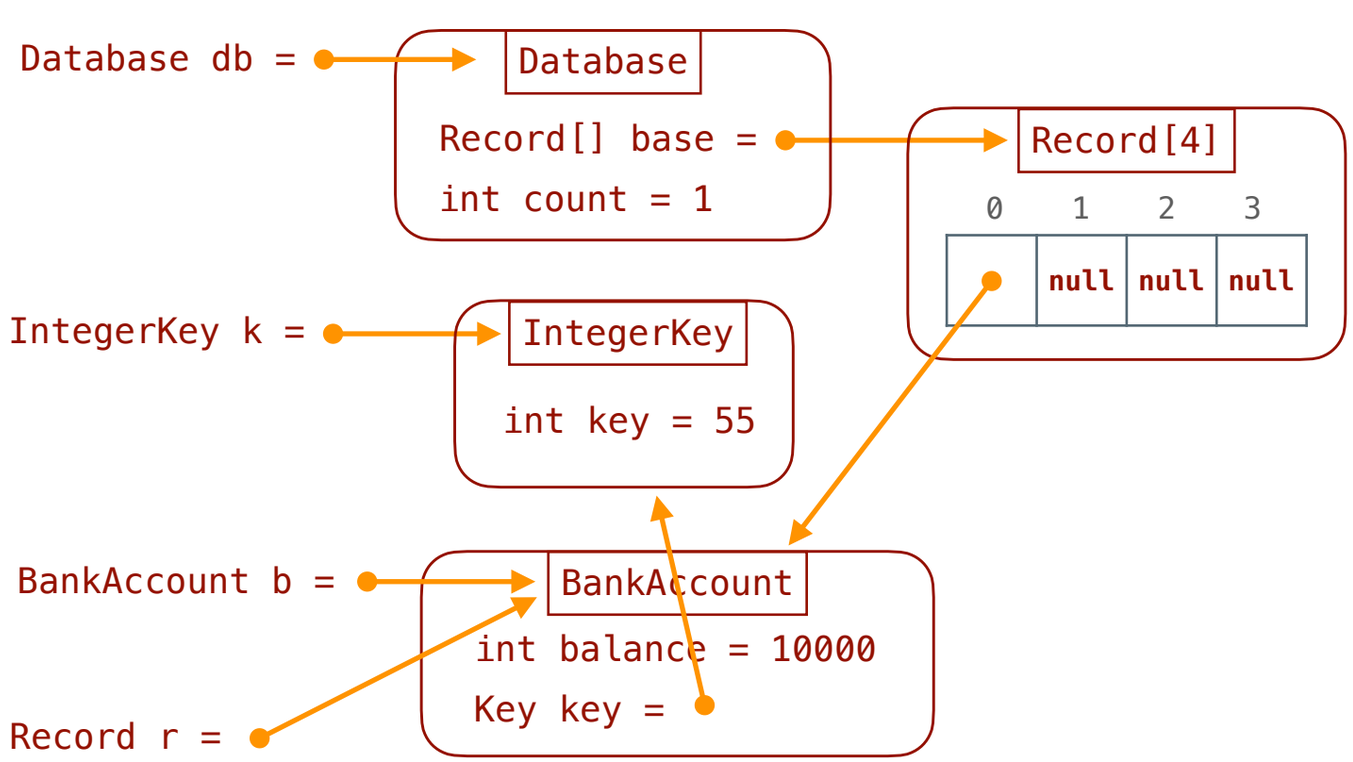


Subtyping

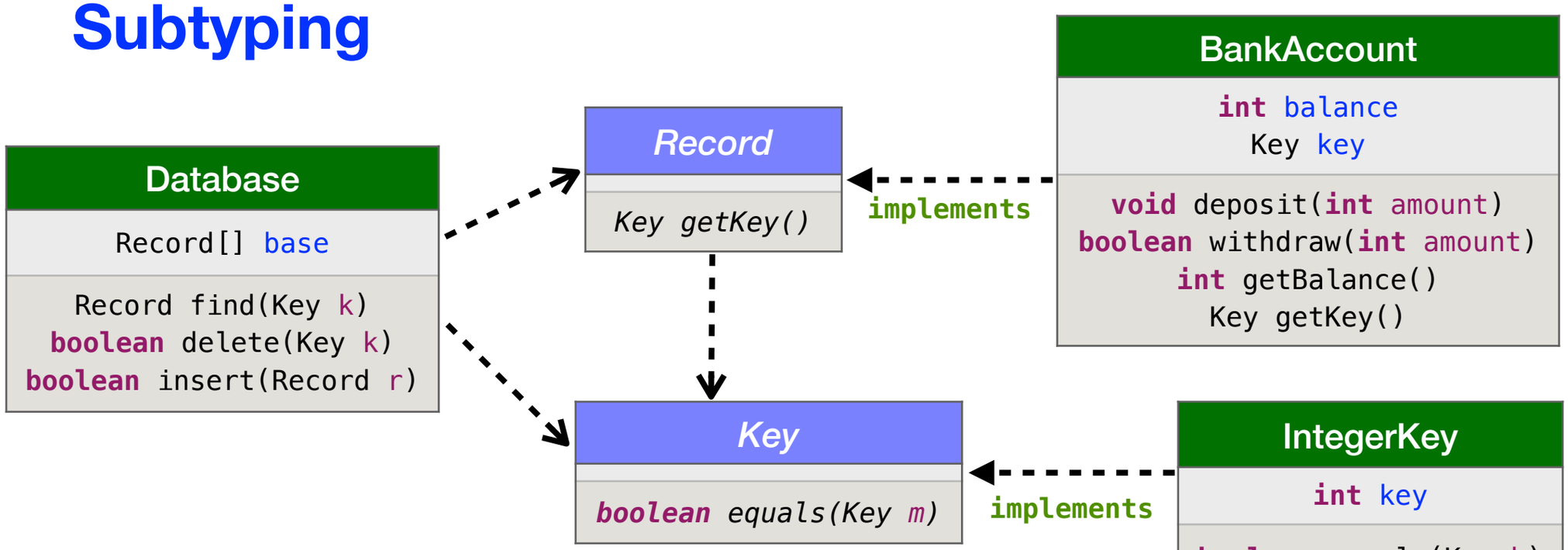



```
public class IntegerKey implements Key
```

```
Record r = db.find(k);

… r.getBalance() …

… ((BankAccount)r).getBalance() …;
```



Subtyping

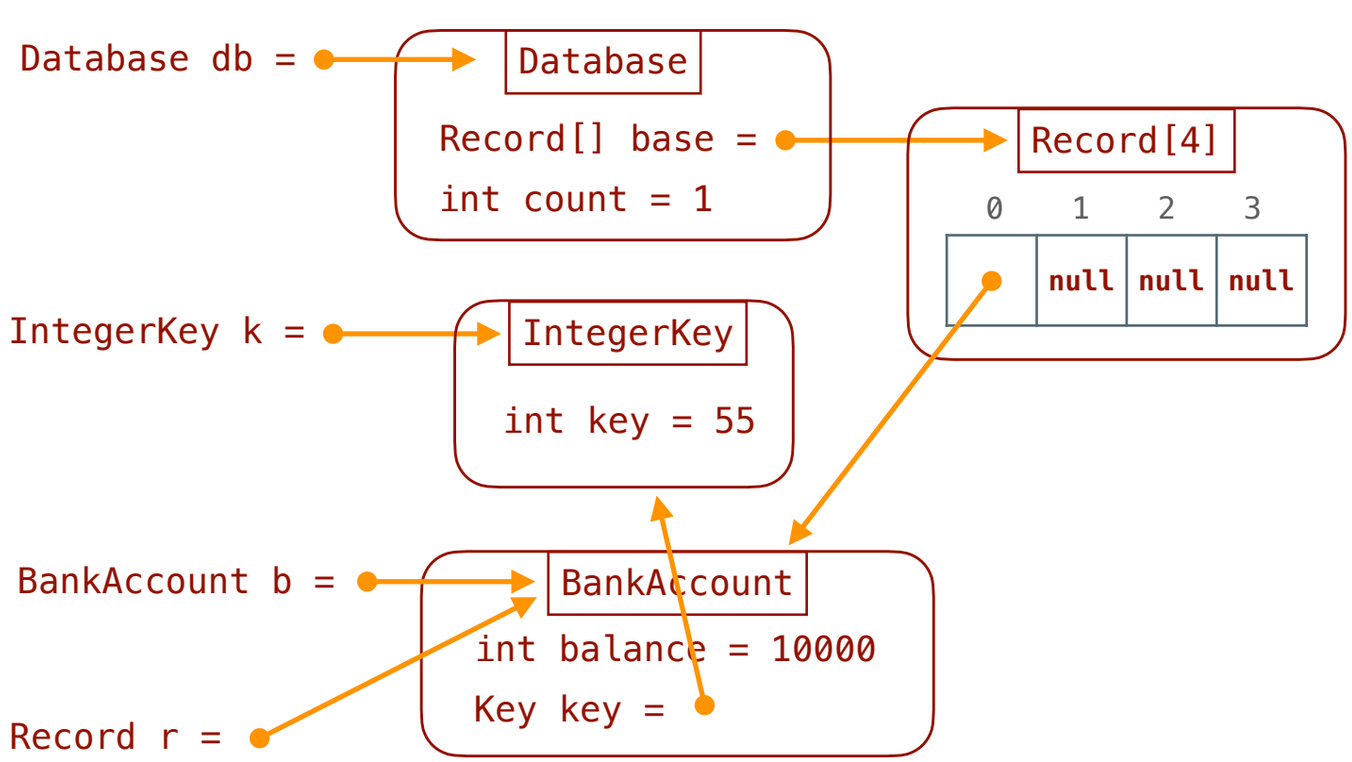



```
public class IntegerKey implements Key
```

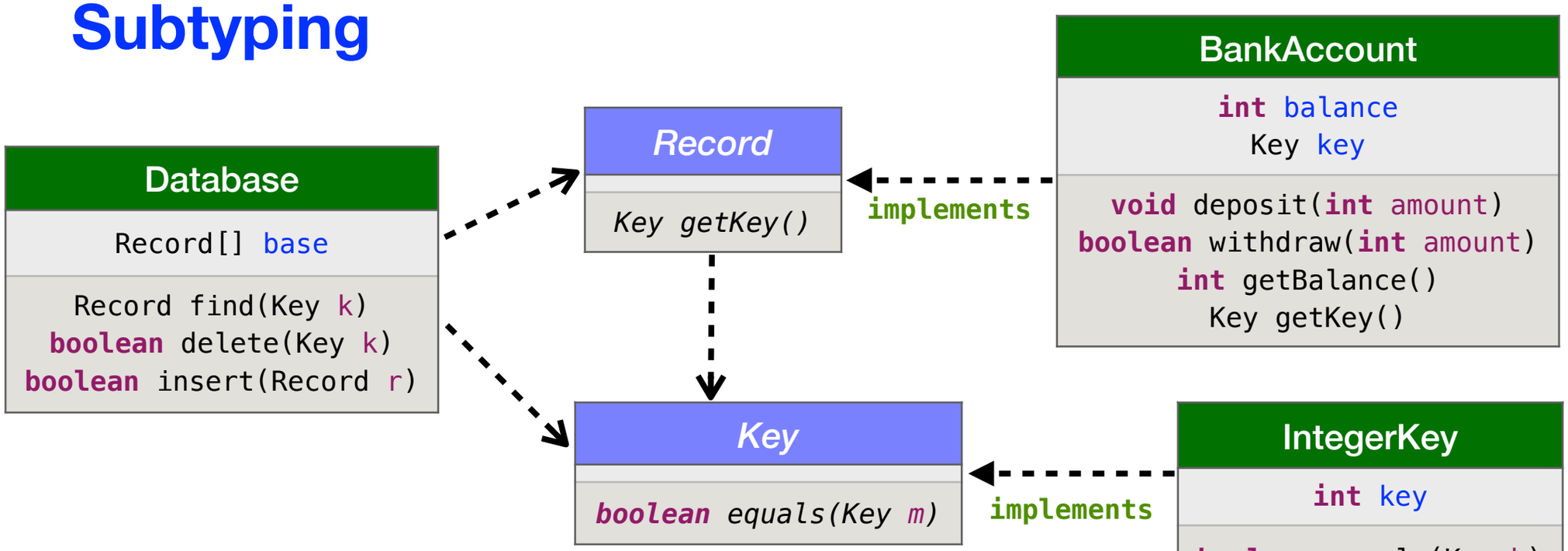
```
Record r = db.find(k);

… r.getBalance() …

… ((BankAccount)r).getBalance() …;
```

컴파일 실패!

타입오류 – Record 인터페이스는 getBalance() 메소드를 모른다.

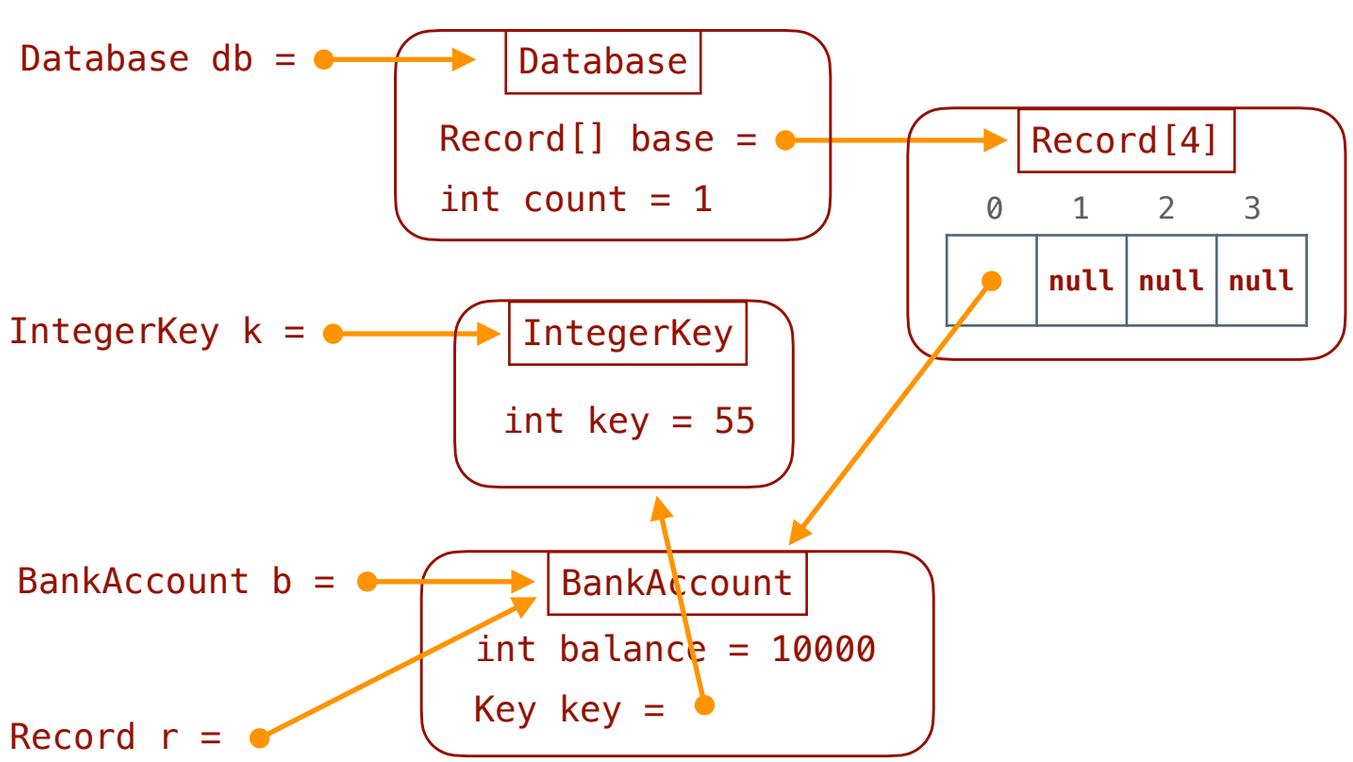


Subtyping

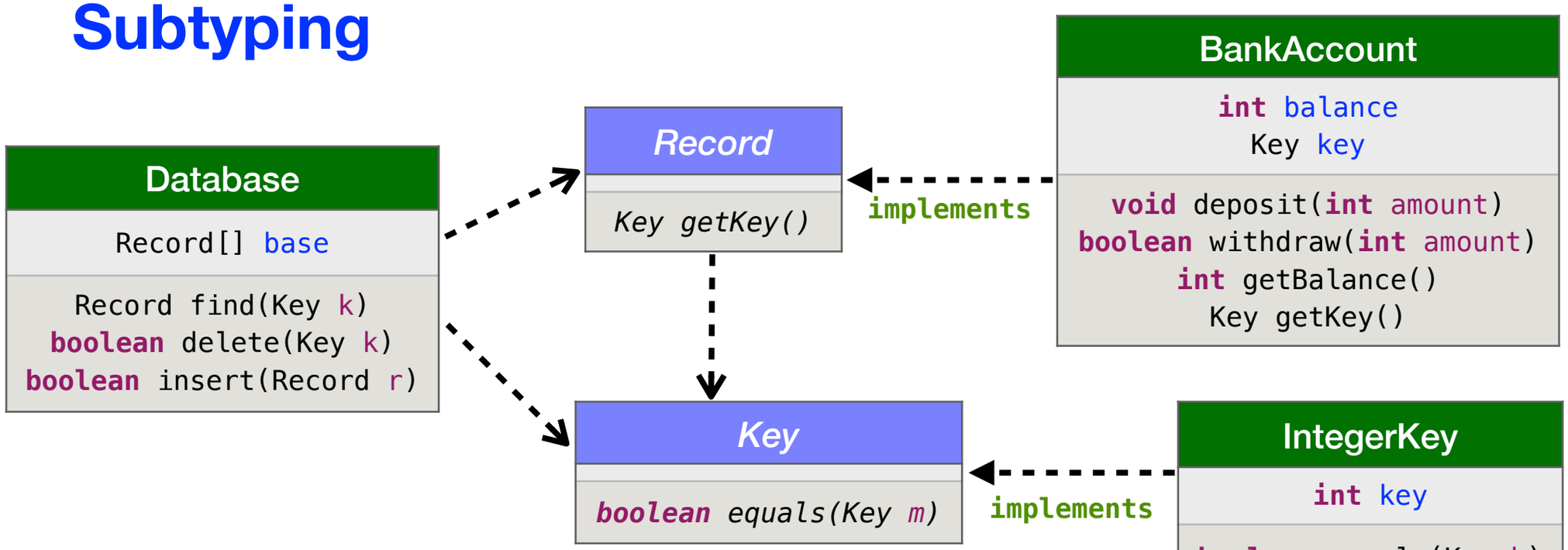



```
public class IntegerKey implements Key
```

```
Record r = db.find(k);

… r.getBalance() …

… ((BankAccount)r).getBalance() …;
```



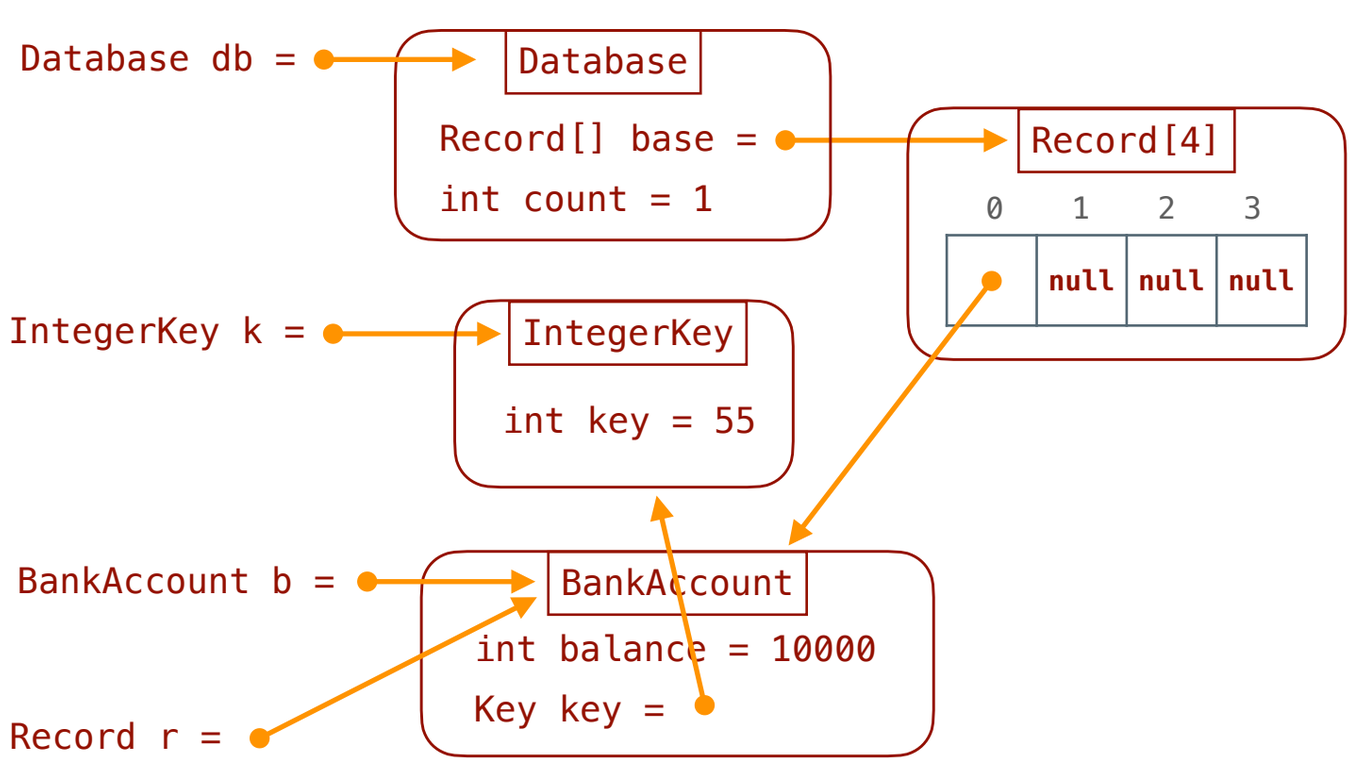
Subtyping



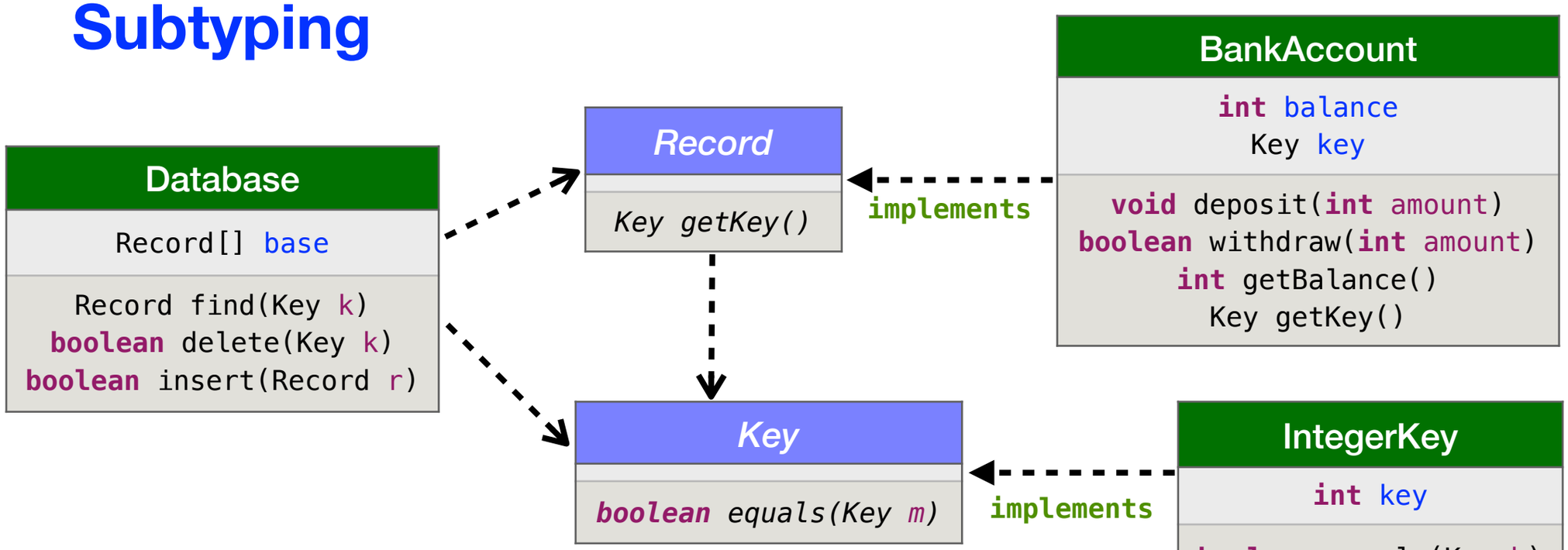

```
public class IntegerKey implements Key
```

```
Record r = db.find(k);

… r.getBalance() …

… ((BankAccount)r).getBalance() …;
```

타입 캐스트하여 컴파일 성공



Subtyping



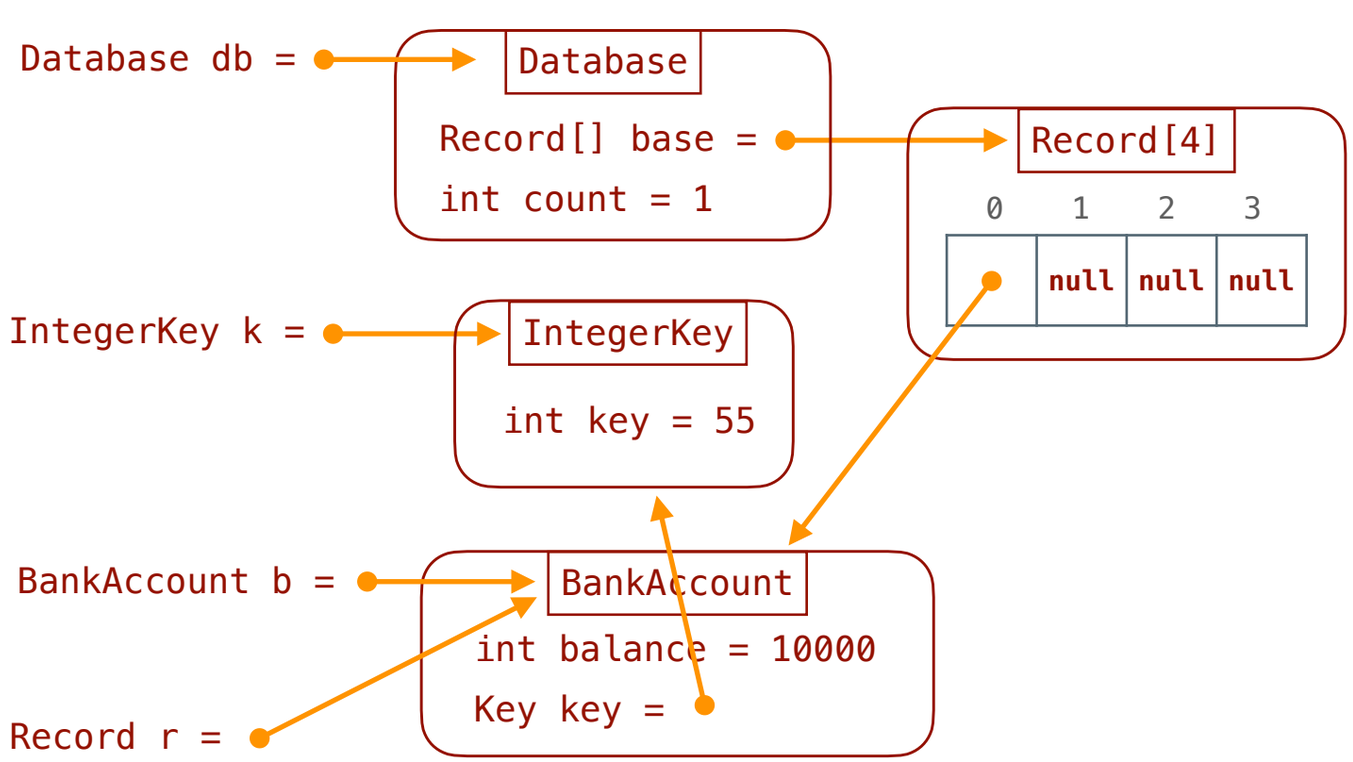

```
public class IntegerKey implements Key
```

BankAccount가 아니라면?

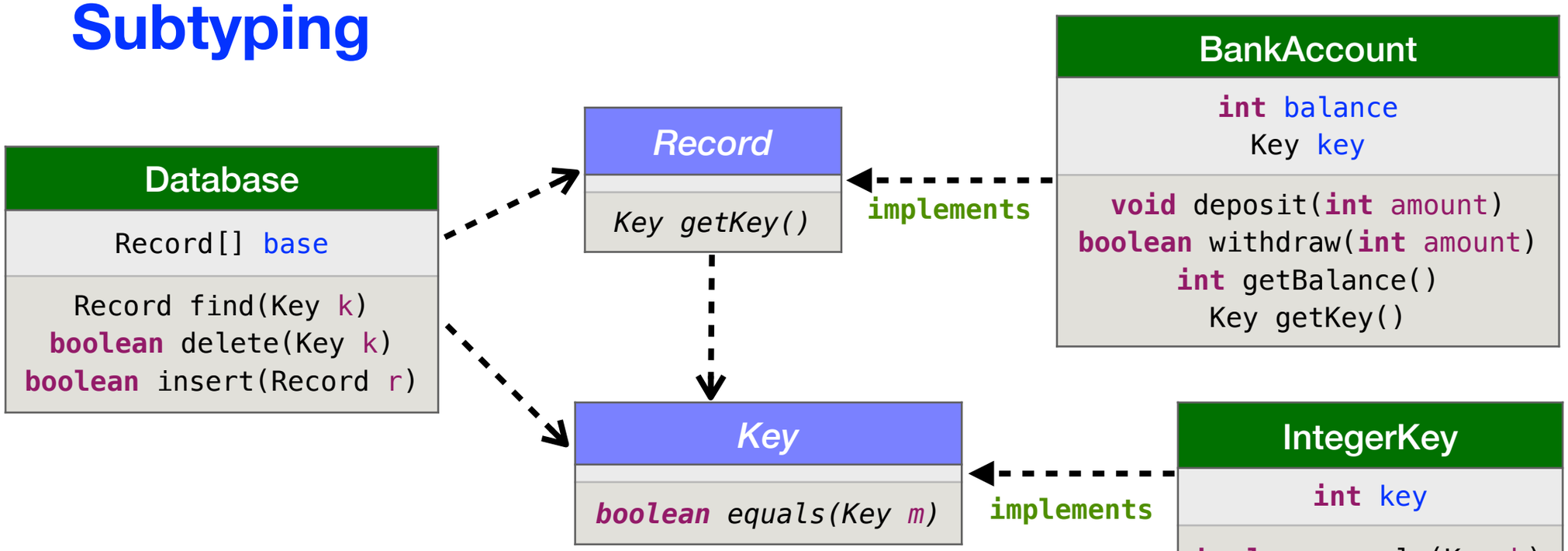
IntegerKey 아니라면?

```
Record r = db.find(some_key);
… ((BankAccount)r).getBalance() … ;
```

컴파일러 통과,
하지만 실행 오류 발생



Subtyping

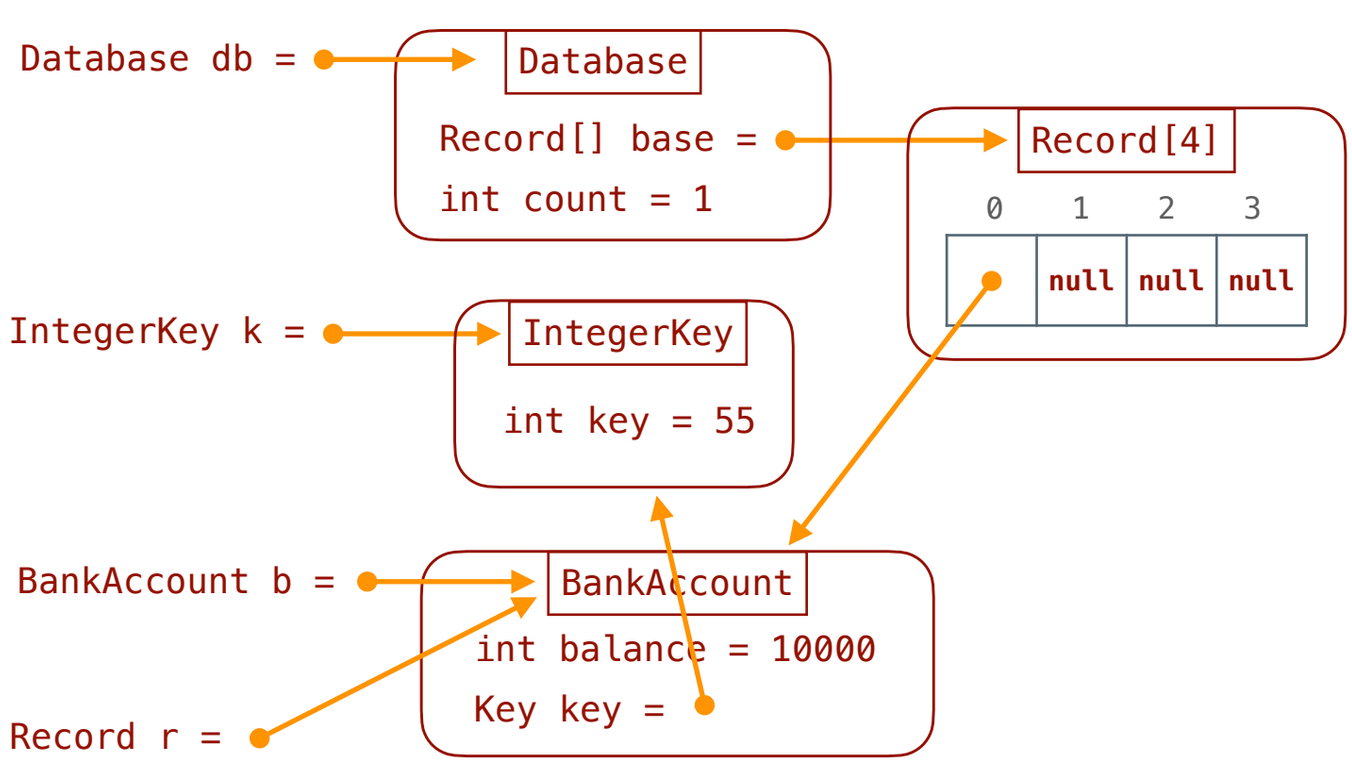



```
public class IntegerKey implements Key
```

```
Record r = db.find(k);

… r.getBalance() …

… ((BankAccount)r).getBalance() …;
```

```
Record r = db.find(some_key);
… ((BankAccount)r).getBalance() … ;
if (r instanceof BankAccount)
    … ((BankAccount)r).getBalance() … ;
else
    System.out.println("취급 불가 Record");
```

r이 BankAccount 인지 사전 확인해야 함



추가 예제

```java
public interface Key {

    /** equals - 인수로 제공된 키와 자신과 같은지 비교
     * @param - 비교 대상 키
     * @return - 같으면 true, 다르면 false */
    public boolean equals(Key m);
}
```

```java
public class StringKey implements Key {
    private String s;

    public StringKey(String s0) {
        s = s0;
    }

    public boolean equals(Key m) {
        return s == ((StringKey)m).getString();
    }

    public String getString() {
        return s;
    }
}
```

```java
IntegerKey k1 = new IntegerKey(3);
StringKey k2 = new StringKey("three");
boolean answer = k2.equals(k1);
```

컴파일러 무사 통과, 그러나 실행중 오류 발생 !

IntegerKey 객체는 getString() 메소드가 없음

수리 방법

```
public class StringKey implements Key {
    private String s;

    public StringKey(String j) {
        s = j;
    }

    public boolean equals(Key m) {
        if (m instanceof StringKey)
           return s.equals(((StringKey)m).getString());
        else
           return false;
    }

    public String getString() {
        return s;
    }
}
```

```
IntegerKey k1 = new IntegerKey(3);
StringKey k2 = new StringKey(“three”);
boolean answer = k2.equals(k1);
```

컴파일 OK,
실행 OK - false 리턴

실습#1. 서브 타입 이해하기

```java
public class Person {
    private String name;

    public Person(String n) {
        name = n;
    }

    public String getName() {
        return name;
    }

    public boolean sameName(Person other) {
        return getName().equals(other.getName();
    }
}
```

```java
public class PersonFrom extends Person {
    private String city;

    public PersonFrom(String n, String c) {
        super(n);
        city = c;
    }

    public String getCity() {
        return city;
    }

    public boolean same(PersonFrom other) {
        return sameName(other) &&
                city.equals(other.getCity());
    }
}
```

```java
Person p = new Person("마음");
Person q = new PersonFrom("소리","서울");
```

다음 각 문장을 이해하고, Java 컴파일러를 통과하는 문장을 고르고, 그 문장이 무엇을 프린트할지 예측해보자.

- `System.out.println(p.sameName(q));`
- `Person x = q; System.out.println(x.getName());`
- `PersonFrom x = p; System.out.println(x.getCity());`
- `Person x = q; System.out.println(x.getCity());`
- `System.out.println(q.same(p));`

실습#2. instanceof

1. IntegerKey를 구현하자.
2. 다음 코드를 실행하면 어떤 결과가 실행창에 프린트 될까?

```
Database db = new Database(4);

BankAccount a1 = new BankAccount(50000, new IntegerKey(55));
Key k = new StringKey("열려라");
BankAccount a2 = new BankAccount(10000, k);
boolean transaction1 = db.insert(a1);
boolean transaction2 = db.insert(a2);


Record p = db.find(k);
BankAccount q = (BankAccount)p;
System.out.println(q.getBalance());

Key k = q.getKey();
if (k instanceof IntegerKey)
    System.out.println(((IntegerKey)k).getInt());
else if (k instanceof StringKey)
    System.out.println(((StringKey)k).getString());
else
    System.out.println("모르는 Key 출현 오류");
```

실습#3. Dealer 클래스 구현

```
1  public interface CardPlayerBehavior {
2
3      /** wantsACard - 카드 한 장을 받겠는지 답한다.
4       * @return 카드를 받고 싶으면 true, 아니면 false */
5      public boolean wantsACard();
6
7      /** receiveCard - 카드 한 장을 받아서 손에 넣는다.
8       * @return 카드 수령 성공이면 true, 실패이면 false */
9      public boolean receiveCard(Card c);
10 }
```

class	Dealer	카드 딜러
method	`void dealTo(CardPlayerBehavior p)`	카드를 한 장씩 매번 물어보면서 원하는 만큼 p에게 준다.
	`void dealOneTo(CardPlayerBehavior p)`	카드를 한 장 p에게 준다.
collaborators	CardPlayerBehavior, CardDeck, Card	

Abstract Class

일부 메소드의 몸체가 비어있는 클래스

```
public abstract class Person {
    private String name;

    public Person(String n) {
        name = n;
    }

    public String getName() {
        return name;
    }

    public abstract String getAddress();

    ……
}
```

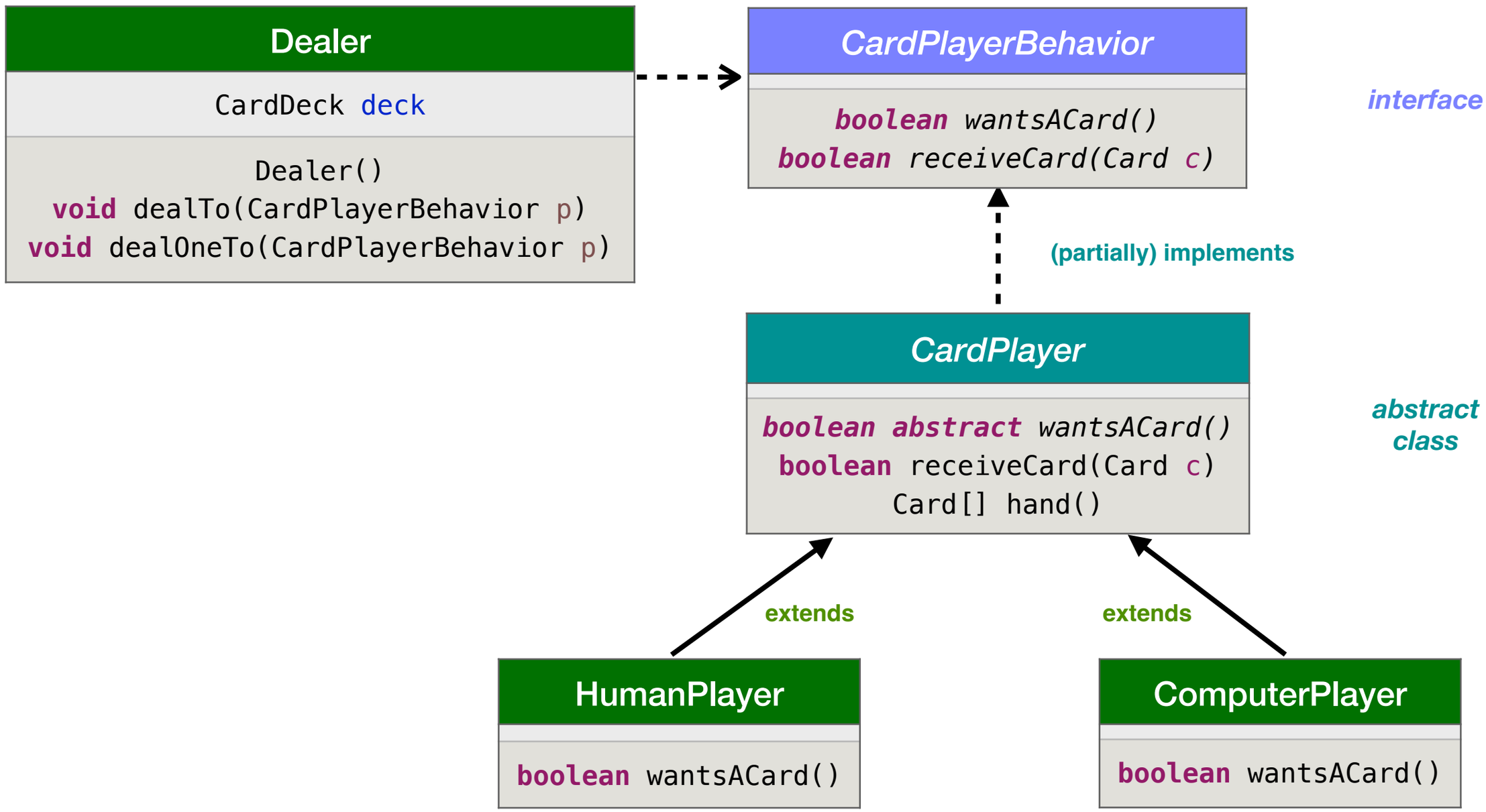
- `new Person()` 불가!
- `extends` 가능

```
public class PersonAddress extends Person {
    private String address;

    public PersonAddress(String n, String a) {
        super(n);
        address = a;
    }

    public String getAddress() {
        return address;
    }

    ……
}
```

```
public class PersonAddrInt extends Person {
    private int address;

    public PersonAddrInt(String n, int a) {
        super(n);
        address = a;
    }

    public String getAddress() {
        return "" + address;
    }

    ……
}
```

실습#4. Card Players 구현

Architecture for Dealer and Card Players



```
ComputerPlayer p = new ComputerPlayer(3);
HumanPlayer h = new HumanPlayer(3);
CardPlayer player1 = p;
CardPlayerBehavior player2 = player1;
```

```
if (player2 instanceof CardPlayer) {
    ((CardPlayer)player2).showcards();
}
```

ComputerPlayer p ==

HumanPlayer h ==

CardPlayer player1 ==

CardPlayerBehavior player2 ==

Card[3]

ComputerPlayer

```
public boolean wantsACard { . . . }

// from CardPlayer
Card[] hand ==
int card_count == 0
public boolean receiveCard(Card c) { . . . }
public Card[] showCards() { . . . }
```

HumanPlayer

```
public boolean wantsACard { . . . }

// from CardPlayer
Card[] hand ==
int card_count == 0
public boolean receiveCard(Card c) { . . . }
public Card[] showCards() { . . . }
```

Card[3]

```
ComputerPlayer
<= CardPlayer
<= CardPlayerBehavior

HumanPlayer
<= CardPlayer
<= CardPlayerBehavior
```

Frameworks

- 프레임워크는 특정 애프리케이션 제작에 특화한 아키텍처 구축용으로 미리 준비하여 모아 놓은 클래스와 인터페이스의 집합체
  - 그래픽 윈도우 구축용 프레임워크
  - 애니메이션 제작용 프레임워크
  - 스프레드쉬트 개발용 프레임워크
  - 음악 작곡용 프레임워크
  - 카드게임 개발용 프레임워크
- 프레임워크의 일부는 `abstract class` 로 비워 둠
- 사례
  - Java’s Abstract Window Toolkit (`java.awt`) package
  - Java’s Swing (`javax.swing`) package

Packages

- 폴더 안에 모아 놓은 클래스와 인터페이스를 통틀어 패키지라고 한다.
  - `java.util`
  - `java.awt`
  - `javax.swing`
- `import <패키지이름>` 의 형식으로 불러쓴다.

class Object

Object는 존재하는 모든 클래스의 최상위 객체

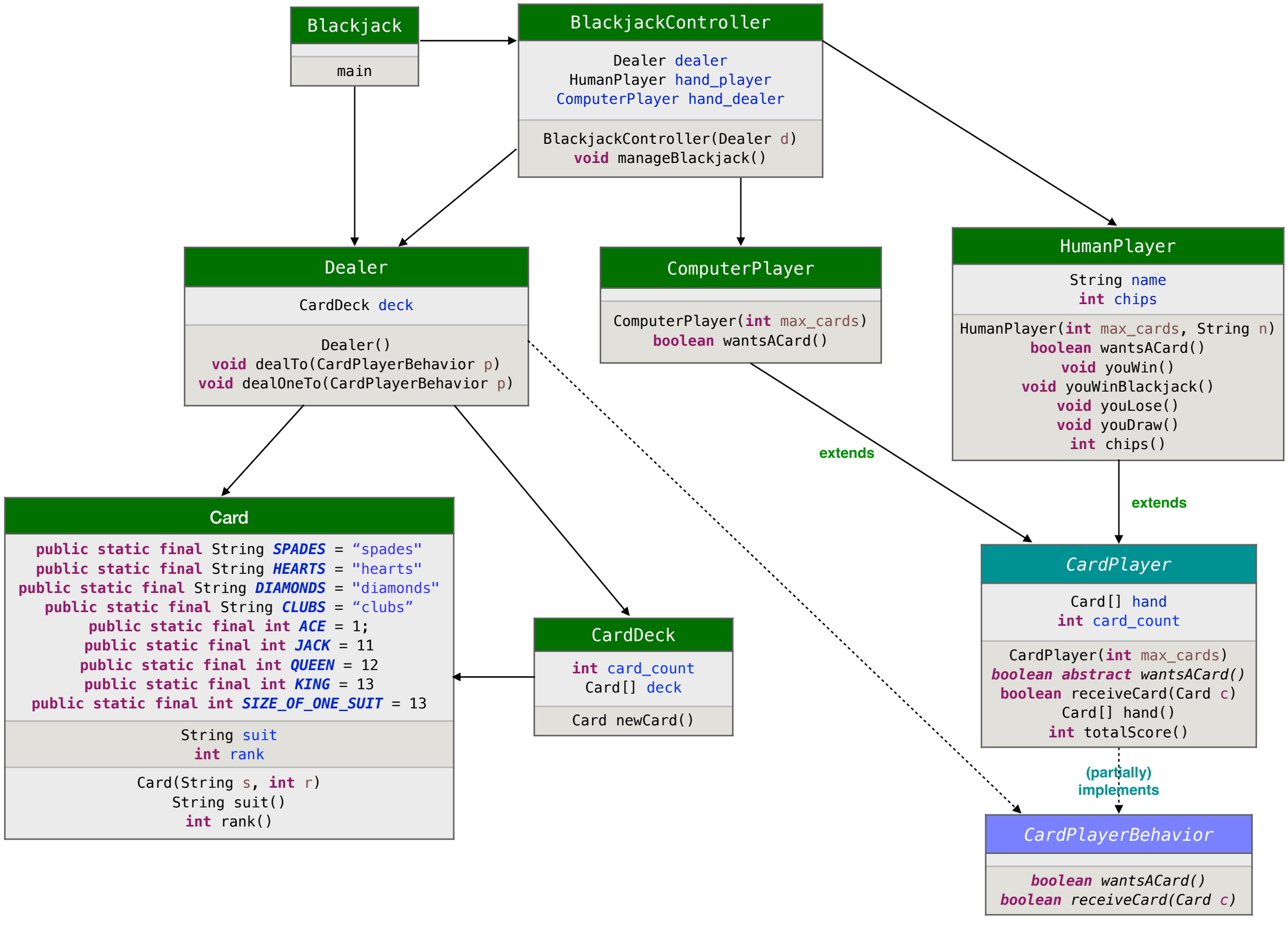
모든 클래스 C에 대해서, C <= Object

소속 패키지: java.lang

```
public class Pair {
    Object[] r = new Object[2];

    public Pair(Object ob1, Object ob2) {
        r[0] = ob1;
        r[1] = ob2;
    }

    public Object get1st(){
        return r[0];
    }

    public Object get2nd(){
        return r[1];
    }
}
```

```
Pair p = new Pair("abc", 7);

Object item1 = p.get1st();
System.out.println((String)item1 + (String)item1);

Object item2 = p.get2nd();
System.out.println((int)item2 + 2);
```

Blackjack


Blackjack
main
BlackjackController
Dealer dealer
HumanPlayer hand_player
ComputerPlayer hand_dealer
BlackjackController(Dealer d)
void manageBlackjack()
Dealer
CardDeck deck
Dealer()
void dealTo(CardPlayerBehavior p)
void dealOneTo(CardPlayerBehavior p)
ComputerPlayer
ComputerPlayer(int max_cards)
boolean wantsACard()
HumanPlayer
String name
int chips
HumanPlayer(int max_cards, String n)
boolean wantsACard()
void youWin()
void youWinBlackjack()
void youLose()
void youDraw()
int chips()
extends
extends
Card
public static final String SPADES = “spades"
public static final String HEARTS = "hearts"
public static final String DIAMONDS = "diamonds"
public static final String CLUBS = “clubs”
public static final int ACE = 1;
public static final int JACK = 11
public static final int QUEEN = 12
public static final int KING = 13
public static final int SIZE_OF_ONE_SUIT = 13
String suit
int rank
Card(String s, int r)
String suit()
int rank()
CardDeck
int card_count
Card[] deck
Card newCard()
CardPlayer
Card[] hand
int card_count
CardPlayer(int max_cards)
boolean abstract wantsACard()
boolean receiveCard(Card c)
Card[] hand()
int totalScore()
(partially) implements
CardPlayerBehavior
boolean wantsACard()
boolean receiveCard(Card c)