

파워자바(개정3판)

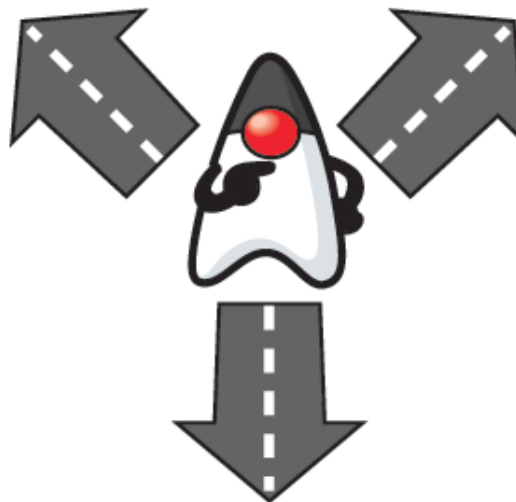


3장 조건문, 반복문, 배열



3장의 목표

1. 조건에 따라 서로 다른 문장을 실행하는 코드를 만들 수 있나요?
2. 조건에 따라서 반복하는 코드를 만들 수 있나요?
3. 자바에서 배열을 생성하고 배열 안에 값을 저장할 수 있나요?
4. 자바에서 2차원 배열을 생성하고 사용할 수 있나요?
5. ArrayList를 생성하고 값을 저장할 수 있나요?



if-else 문

- 조건에 따라서 서로 다른 처리를 하고 싶을 때 사용하는 구조가 if-else 문이다.

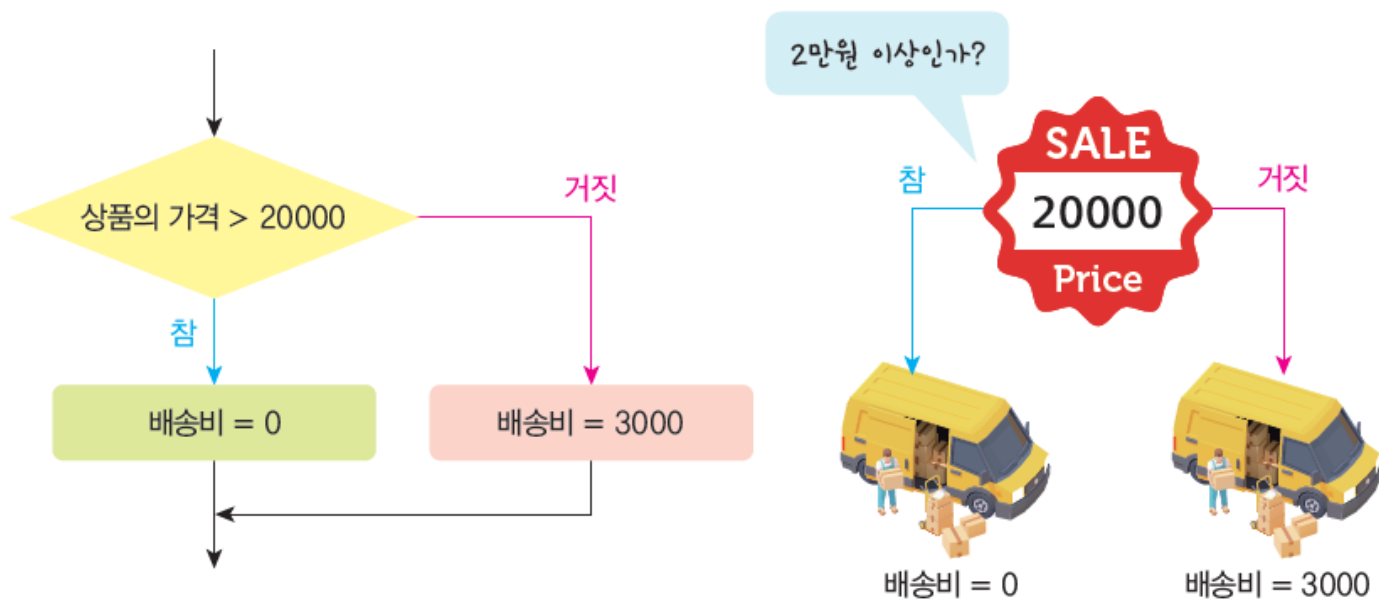


그림 3.1 조건문을 순서도로 그린 것이다.



if-else 문 의 구조

Syntax: if-else 문

```
if (price > 20000)
```

참이나 거짓으로
계산되는 조건식

```
shipping_cost = 0;
```

조건식이 참이면 실행되는 문장

```
else
```

else절은 생략될
수도 있다.

```
shipping_cost = 3000;
```

조건식이 거짓이면 실행되는 문장



예제: 짝수와 홀수 구별하기

키보드에서 입력받은 정수가 홀수인지 짝수인지를 말해주는 프로그램을 작성하여 보자. 홀수와 짝수는 어떻게 구별할 수 있는가? 홀수는 2로 나누었을 때 나머지가 1이다. 짝수는 2로 나누었을 때 나머지가 0이다. 따라서 나머지 연산자 %를 이용하여 구별할 수 있다.

정수를 입력하시오: 2

입력된 정수는 짝수입니다.

실행 결과



예제: 짝수와 홀수 구별하기

```
import java.util.Scanner;

public class EvenOdd {
    public static void main(String[] args) {

        int number;

        Scanner sc = new Scanner(System.in);
        System.out.print("정수를 입력하세요: ");
        number = sc.nextInt();

        if (number % 2 == 0) {
            System.out.println("입력된 정수는 짝수입니다.");
        } else {
            System.out.println("입력된 정수는 홀수입니다.");
        }
    }
}
```



다중 if-else 문

```
public class Nested {  
    public static void main(String[] args) {  
  
        Scanner sc = new Scanner(System.in);  
        System.out.print("정수를 입력하시오: ");  
        int number = sc.nextInt();  
  
        if(number > 0 )  
            System.out.println("양수입니다.");  
        else if (number ==0 )  
            System.out.println("0입니다.");  
        else  
            System.out.println("음수입니다.");  
    }  
}
```

정수를 입력하시오: 10
양수입니다.



예제: 학점 결정

종종 우리는 조건에 따라서 다중으로 분기되는 결정을 내려야 하는 경우가 있다. 학생들의 성적을 받아서 학점을 출력하는 프로그램을 작성하여 실행하여 보자. 성적이 90점 이상이면 A학점, 80점 이상이고 90점 미만이면 B학점, 70점 이상이고 80점 미만이면 C학점과 같이 결정하는 것이다.

성적을 입력하시오: 92

학점 A



예제: 학점 계산

```
import java.util.Scanner;

public class Grading {
    public static void main(String[] args) {
        int score;

        Scanner sc = new Scanner(System.in);

        System.out.print("성적을 입력하시오: ");
        score = sc.nextInt();
        if (score >= 90)                                     // ①
            System.out.println("학점 A");
        else if (score >= 80)                                // ②
            System.out.println("학점 B");
        else if (score >= 70)
            System.out.println("학점 C");
        else if (score >= 60)
            System.out.println("학점 D");
        else
            System.out.println("학점 F");

    }
}
```



예제: 가위, 바위, 보 게임

가위, 바위, 보 게임을 작성하여 보자. 텍스트 버전은 여기서 작성하고 그래픽 버전은 8장에서 작성해보자. 사용자가 가위, 바위, 보 중에서 하나를 선택하면 이것을 컴퓨터가 생성한 난수값과 비교한다. 누가 이겼는지를 화면에 출력한다.



가위(0), 바위(1), 보(2): 1

인간: 1 컴퓨터: 0 인간 승리



예제: 가위, 바위, 보 게임

```
import java.util.*;

public class RockPaperScissor {
    final int SCISSOR = 0;
    final int ROCK = 1;
    final int PAPER = 2;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("가위(0), 바위(1), 보(2): ");
        int user = sc.nextInt();

        int computer = (int) (Math.random() * 3);
        if( user == computer )
            System.out.println("인간과 컴퓨터가 비겼음");
        else if (user == (computer + 1) % 3) // 0은 1한테 지고 1은 2한테, 2는 0한테 진다.
            System.out.println("인간: " + user + " 컴퓨터: " + computer + " 인간 승리");
        else
            System.out.println("인간: " + user + " 컴퓨터: " + computer + " 컴퓨터 승리");
    }
}
```



조건문

1. 자바에서 참과 거짓은 무엇으로 표시되는가?
2. 조건에 따라서 실행되어야 하는 문장이 두 개 이상이면 어떻게 하여야 하는가?
3. 실수와 실수를 비교할 때 주의해야 할 점은 무엇인가?
4. 수식 (!true)의 값은?
5. 다음 코드는 어떤 오류를 가지고 있는가?

```
if (x=1) { System.out.println("x는 1입니다."); }
```
6. 속도가 60km/h 이상이고 100km/h 이하이면 “정상 속도”라고 출력하는 코드를 작성해보자.
7. x와 y 중에서 큰 값을 max에 저장하고 작은 값을 min에 저장하는 코드를 작성해보자.



switch 문

Syntax: switch 문

```
switch(제어식)
```

```
{
```

```
    case c1:  
        문장1;  
        break;
```

제어식의 값이 c1이면 실행된다.

```
    case c2:  
        문장2;  
        break;
```

제어식의 값이 c2이면 실행된다.

```
    ...
```

```
    default:  
        문장d;  
        break;
```

일치하는 값이 없으면 실행된다.

```
}
```



예제: 학점 계산

사용자가 자신의 성적을 입력하면 성적을 학점으로 변환하여 출력해주는 프로그램을 작성하여 보자. 이것은 앞에서 if-else 문장으로도 작성한 바 있다. if-else 문장을 사용하는 방법과 비교하여 보자.

성적을 입력하시오: 92

학점: A



예제: 학점 계산

```
import java.util.*;

public class Score2Grade {
    public static void main(String[] args) {

        int score, number;
        char grade;

        Scanner sc = new Scanner(System.in);
        System.out.print("성적을 입력하시오: ");
        score = sc.nextInt();
        number = score / 10;
        switch (number) {
            case 10:
            case 9:      grade = 'A';      break;
            case 8:      grade = 'B';      break;
            case 7:      grade = 'C';      break;
            case 6:      grade = 'D';      break;
            default:     grade = 'F';      break;
        }
        System.out.print("학점: " + grade);
    }
}
```



switch 문에 문자열 사용

- Java 7부터는 switch 문의 제어식으로 String 객체를 사용할 수 있다.

```
String s = "yes";  
switch(s) {  
    case "yes": ...  
        break;  
    case "no": ...  
        break;  
}
```




예제:

피자 종류를 받아서 피자의 가격을 반환하는 프로그램을 작성해보자.

피자 종류를 입력하시오: 콤비네이션

피자 콤비네이션의 가격=20000



예제:

```
import java.util.Scanner;

public class StringSwitch {
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.print("피자 종류를 입력하시오. ");

        String model = sc.next();
        int price=0;

        switch (model) {
            case "콤비네이션":
            case "슈퍼슈프림": price = 20000; break;
            case "포테이토": price = 15000; break;
            case "쉬림프": price = 25000; break;
            default: price = 0; break;
        }
        System.out.println("피자 "+model+"의 가격은"+price);
    }
}
```



향상된 switch 문

- Java 12부터는 “화살표”를 사용하는 향상된 switch 문을 사용할 수 있다. “case->S”과 같은 형식을 사용한다.

```
public class Test {  
    public static void main(String[] args) {  
        var day = "SAT";  
        var today = "";  
        switch (day) {  
            case "SAT", "SUN" -> today = "주말";  
            case "MON", "TUS", "WED", "THU", "FRI" -> today = "주중";  
            default -> System.out.println("Error");  
        }  
        System.out.println(today);  
    }  
}
```

주말



중간 점검

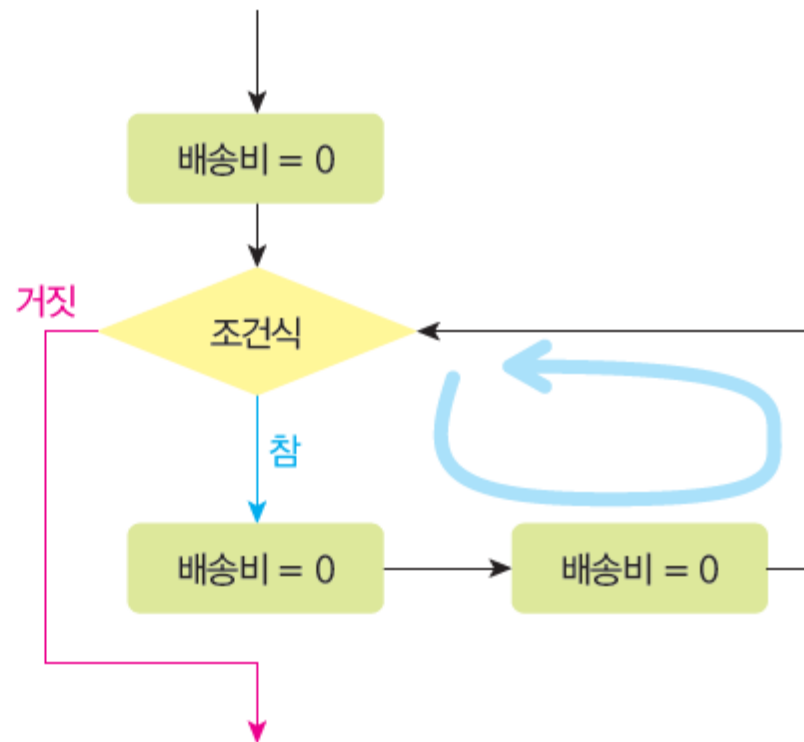
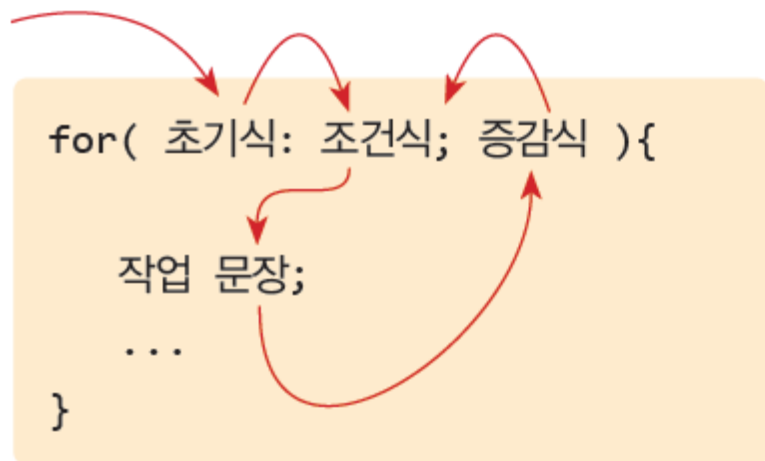
1. 다음 중에서 switch 문의 제어식으로 사용할 수 없는 것은?
① 정수 10 ② 문자 'a' ③ 문자열 "abc" ④ 실수 3.14
2. 자바에서 다중 선택 구조를 만드는 2가지 방법을 이야기해보자.
3. x=1, y=2, z=3일 때, 다음 코드를 수행한 후의 x, y, z의 값은 얼마인가?

```
switch(x) {  
    case 0: z=x+1; break;  
    case 1: y=z+x; break;  
    default: z=z+x;  
}
```

4. 3번 코드를 →을 사용하는 새로운 switch 구문으로 작성해보자.

for 문

```
for( 초기식: 조건식; 증감식 ){  
    작업 문장;  
    ...  
}
```





for 문

Syntax: for 문

초기식

조건식

증감식

```
for( int i=0; i<10; i++ ) {  
    printf("Hello World!");  
}
```

반복되는 문장, 콘솔에 "Hello World!"가 10번 반복된다.



예제: 0부터 4까지 출력하기

다음 프로그램은 for를 사용하여 0부터 4까지의 숫자를 화면에 출력한다.

```
i의 값은: 0
```

```
i의 값은: 1
```

```
i의 값은: 2
```

```
i의 값은: 3
```

```
i의 값은: 4
```



예제: 0부터 4까지 출력하기

```
public class ForExample1 {  
  
    public static void main(String[] args) {  
        for (int i = 0; i < 5; i++) {  
            System.out.println("i의 값은: " + i);  
        }  
    }  
}
```




예제: 정수의 합 계산하기

1부터 10까지의 정수의 합 = 55

```
public class Sum {  
    public static void main(String[] args) {  
  
        int sum = 0;  
  
        for (int i = 1; i <= 10; i++)  
            sum += i;  
  
        System.out.printf("1부터 10까지의 정수의 합 = %d\n", sum);  
    }  
}
```



예제: 팩토리얼 계산하기

이번 예제에서는 팩토리얼 값을 계산하여 보자. 팩토리얼이란 다음과 같이 정의된다.

$$n! = 1 \times 2 \times 3 \times \dots \times n$$

정수를 입력하시요: 20

20!은 2432902008176640000입니다.



예제: 팩토리얼 계산하기

```
import java.util.*;

public class Factorial {
    public static void main(String[] args) {
        long fact = 1;
        int n;

        System.out.printf("정수를 입력하세요:");

        Scanner scan = new Scanner(System.in);
        n = scan.nextInt();

        for (int i = 1; i <= n; i++)
            fact = fact * i;

        System.out.printf("%d!은 %d입니다.\n", n, fact);
    }
}
```



예제: 약수 계산하기

양의 정수를 입력하시오: 100
100의 약수는 다음과 같습니다.
1 2 4 5 10 20 25 50 100



예제: 약수 계산하기

```
import java.util.Scanner;

public class Divisor {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.print("양의 정수를 입력하십시오: ");
        int n = scan.nextInt();

        System.out.println(n + "의 약수는 다음과 같습니다.");
        for (int i = 1; i <= n; ++i) {
            if (n % i == 0)
                System.out.print(" " + i);
        }
    }
}
```



주간 점검

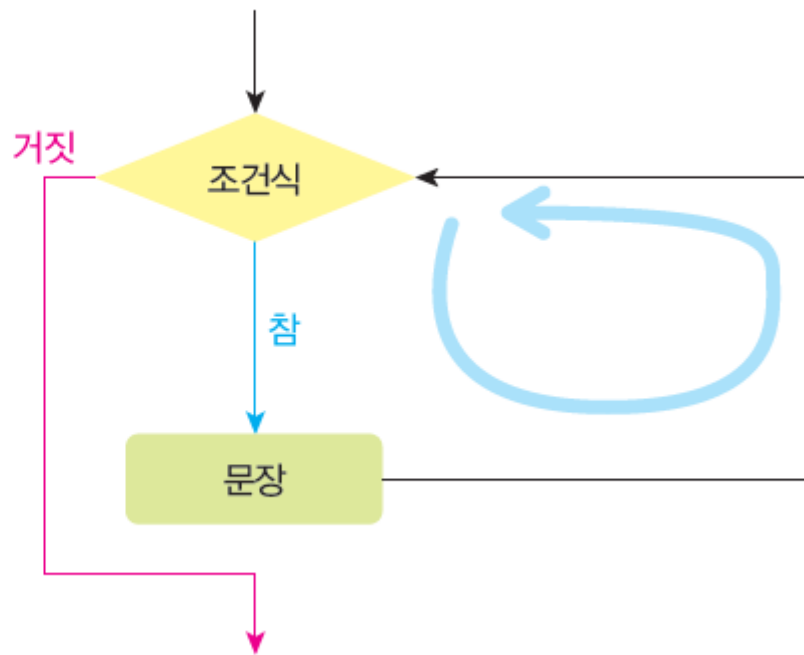
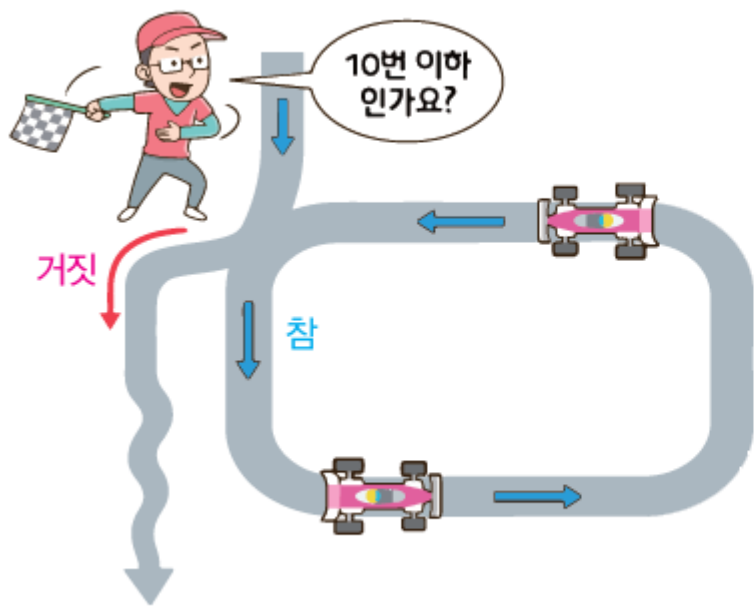
1. 다음 중에서 switch 문의 제어식으로 사용할 수 없는 것은?

① 정수 10 ② 문자 'a' ③ 문자열 "abc" ④ 실수 3.14

2. 자바에서 다중 선택 구조를 만드는 2가지 방법을 이야기해보자.



while 문





while 문

Syntax: while 문

조건식

```
while( i < 10 )  
    printf("Hello World!\n");
```

조건식이 참이면 문장을
반복실행한다.



예제: "환영합니다."를 화면에 5번 출력하는 예제

```
public class WelcomeLoop {  
    public static void main(String[] args) {  
        int i = 0;  
        while (i < 5) {  
            System.out.println("환영합니다!");  
            i++;  
        }  
        System.out.println("반복이 종료되었습니다.");  
    }  
}
```

```
환영합니다!  
환영합니다!  
환영합니다!  
환영합니다!  
환영합니다!  
반복이 종료되었습니다.
```



예제: -1의 값이 입력될 때까지 합계 계산하기

while 문은 조건으로 반복할 때 사용된다. -1의 값이 입력될 때까지, 사용자가 입력한 정수의 합계를 계산하여 출력하는 프로그램을 작성해보자.

정수를 입력하시오: 10

정수를 입력하시오: 20

정수를 입력하시오: 30

정수를 입력하시오: -1

정수의 합은 60입니다.



예제: -1의 값이 입력될 때까지 합계 계산하기

```
import java.util.Scanner;

public class GetSum {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);

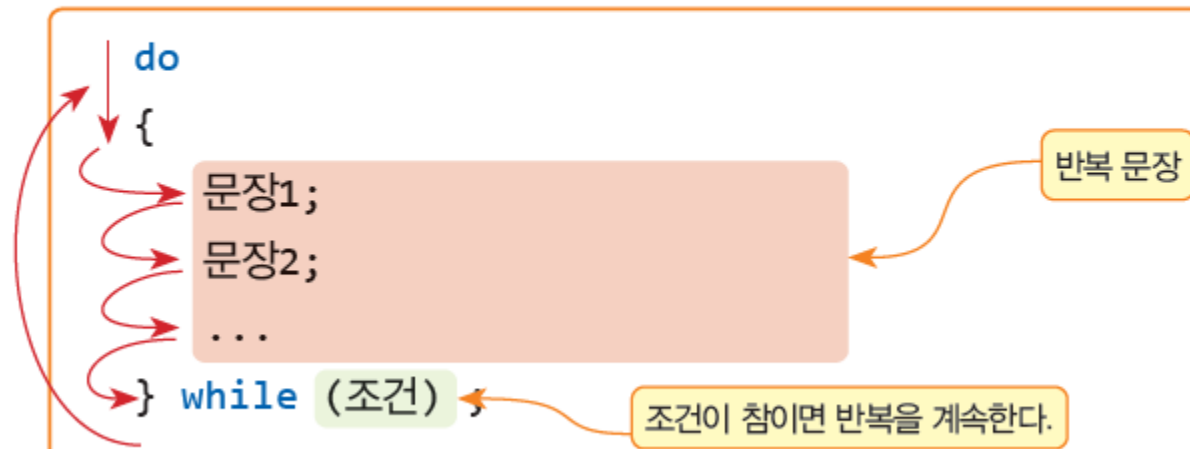
        int sum = 0, value = 0;

        while (value != -1) {
            sum = sum + value;
            System.out.print("정수를 입력하시오: ");
            value = sc.nextInt();
        }
        System.out.println("정수의 합은 " + sum + "입니다.");
    }
}
```



do-while문

Syntax: while 문





예제: 정확한 입력받기

사용자로부터 월의 번호를 입력받는 프로그램을 작성하여 보자. 사용자가 올바른 월 번호를 입력할 때까지 반복을 계속한다. 사용자가 올바른 월 번호를 입력해야만 다음 문장으로 넘어간다.

올바른 월을 입력하십시오 [1-12]: 13

올바른 월을 입력하십시오 [1-12]: 14

올바른 월을 입력하십시오 [1-12]: 0

올바른 월을 입력하십시오 [1-12]: 1

사용자가 입력한 월은 1



예제: 정확한 입력받기

```
import java.util.Scanner;

public class CheckInput {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int month;
        do {
            System.out.print("올바른 월을 입력하시오 [1-12]: ");
            month = sc.nextInt();
        } while (month < 1 || month > 12);
        System.out.println("사용자가 입력한 월은 " + month);
    }
}
```

```
올바른 월을 입력하시오 [1-12]: 13
올바른 월을 입력하시오 [1-12]: 14
올바른 월을 입력하시오 [1-12]: 0
올바른 월을 입력하시오 [1-12]: 1
사용자가 입력한 월은 1
```



중첩 반복문

- 반복문은 중첩되어 사용될 수 있다. 즉 반복문 안에 다른 반복문이 실행될 수 있다. 이러한 형태를 중첩 반복문(nested loop)이라고 한다.

```
for(int i=0; i<5; i++) {  
    for(int k=0; k<5; k++) {  
        반복문장;  
    }  
}
```

외부 반복문

내부 반복문



예제: 사각형 모양 출력하기

- 중첩 반복문은 실제 프로그래밍에서 많이 나오는 형태로 특히 사각형과 비슷한 데이터를 처리하는데 유용하다. 다음 예제는 *기호를 사각형 모양으로 출력한다.

```
*****  
*****  
*****  
*****  
*****
```




예제: 사각형 모양 출력하기

```
import java.util.*;

public class NestedLoop {
    public static void main(String[] args) {

        for (int y = 0; y < 5; y++) {
            for (int x = 0; x < 10; x++)
                System.out.print("*");

            System.out.println("");
        }
    }
}
```



break, continue

```
01 for(int i=1; i<6; i++) {  
02     if(i==4)  
03         break; // 4이면 반복문을 벗어난다.  
04     System.out.println(i);  
05 }
```

1 2 3

```
01 for(int i=1; i<6; i++) {  
02     if(i==4)  
03         continue; // 4일 때는 출력하지 않고 다음 반복으로 넘어간다.  
04     System.out.println(i);  
05 }
```

1 2 3 5 6



무한루프

- **while** 문을 사용할 때, 종료 조건을 만들려면 상당히 까다로운 경우가 있다->이 경우에는 **while(true)**를 이용하여 무한 루프를 만들고 무한 루프 안에서 **break**를 사용하여서 루프를 빠져나가는 조건을 기술하는 편이 가독성이 높고 코딩하기 쉽다

```
while( c != 'q' || count <= 100 ) {  
    ...  
}
```



```
while( true ) {  
    if( c == 'q' ) break;  
    if( count > 100 ) break;  
    ...  
}
```

고재 오타



예제:

- 사용자가 입력한 점수들의 평균을 내는 프로그램을 작성한다. 만약 사용자가 음수를 입력하면 **break**에 의하여 반복 루프가 종료되게 한다.

```
점수를 입력하시오: 10  
점수를 입력하시오: 20  
점수를 입력하시오: -1  
평균은 15
```



예제:

```
import java.util.*;

public class Averager {
    public static void main(String[] args) {
        int total = 0, count = 0;
        Scanner sc = new Scanner(System.in);

        while (true) {
            System.out.print("점수를 입력하시오: ");
            int grade = sc.nextInt();
            if (grade < 0)                break;
            total += grade;
            count++;
        }
        System.out.println("평균은 " + total / count);
    }
}
```



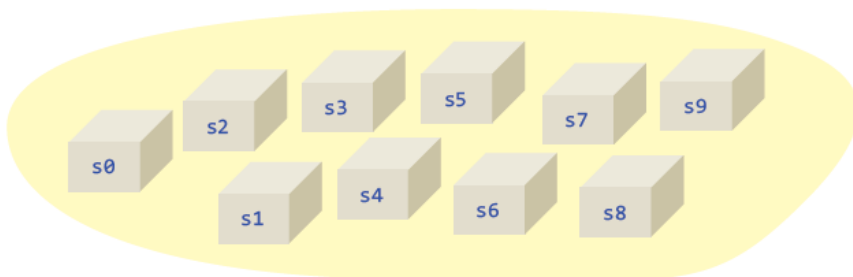
중간점검

1. do-while 문에서 반복 조건이 거짓이면 몇 번이나 반복되는가?
2. do-while을 이용하여 사용자가 'q'를 입력할 때까지 정수의 합계를 계산하는 프로그램을 작성해보자.
3. 중첩 반복문을 이용하여서 구구단을 2단부터 9단까지 출력하여 보자.
4. `for(;;) i=i+1;` 은 몇 번이나 반복하는가?
5. 1부터 100 사이의 정수에서 6의 배수의 합을 계산하는 프로그램을 작성해보자.

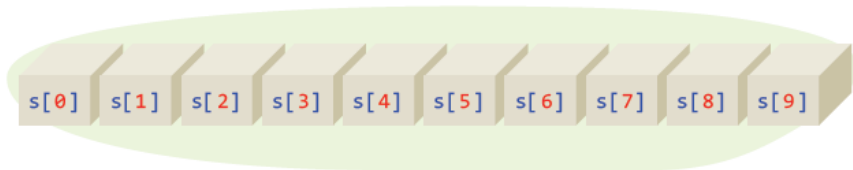


배열

- 배열(array)은 여러 개의 변수를 하나로 묶어 넣은 것이다. 배열을 사용하면 같은 종류의 대량의 데이터를 한 번에 선언할 수 있다.



배열은 변수들을 모아놓은 것
배열은 하나의 이름을 공유한다.



```
int[] s = new int[10];
```

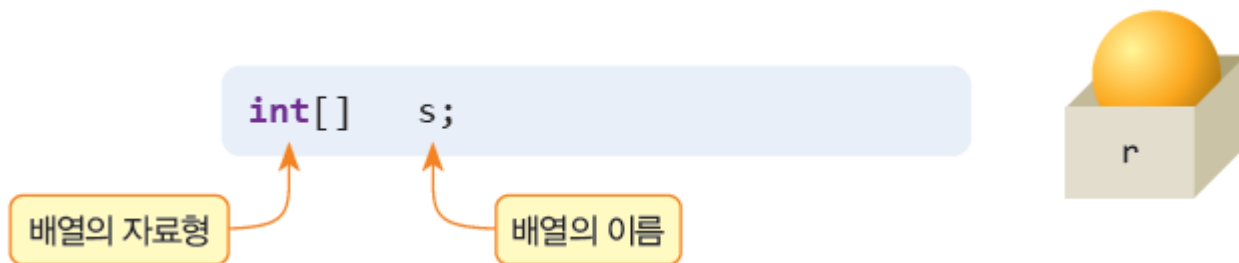
배열은 동일한 타입의 데이터를 여러 개
저장할 수 있는 저장 장소입니다.





배열의 선언과 사용

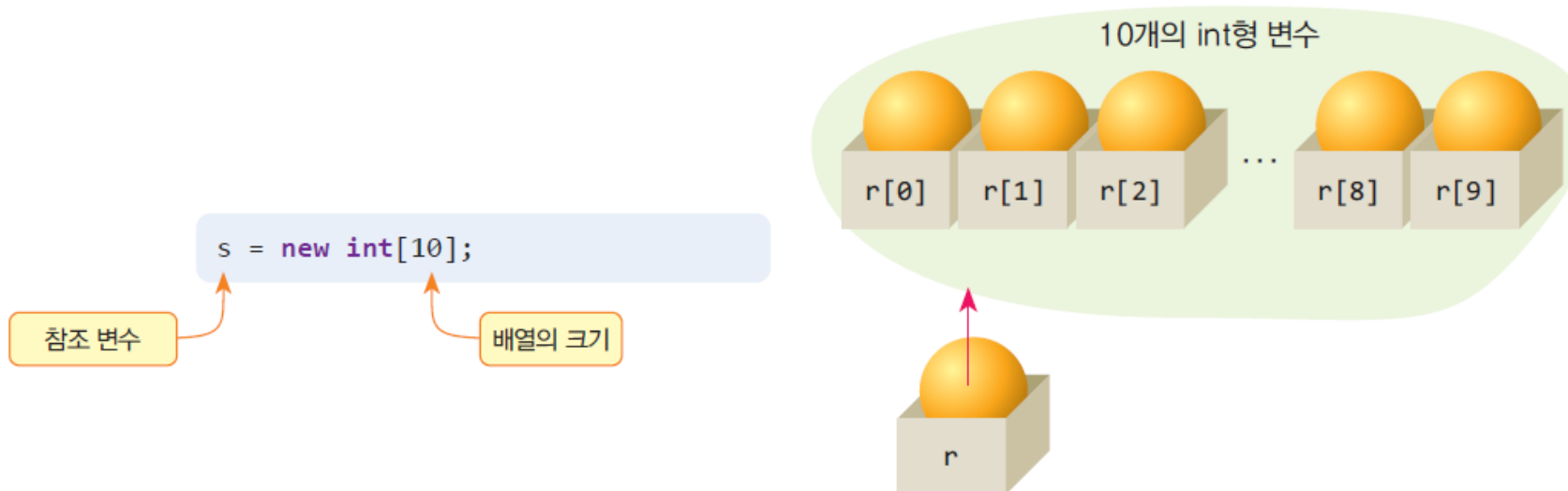
- 배열 참조 변수부터 다음과 같이 선언한다. 정수형 배열을 만든다면 배열 참조 변수는 `int[]` 타입으로 선언하면 된다.





배열의 선언과 사용

- 실제 배열은 **new** 연산자를 사용하여 생성하여야 한다.





반복문과 배열

- 배열의 각각의 요소는 인덱스(index)라는 번호로 접근할 수 있다.

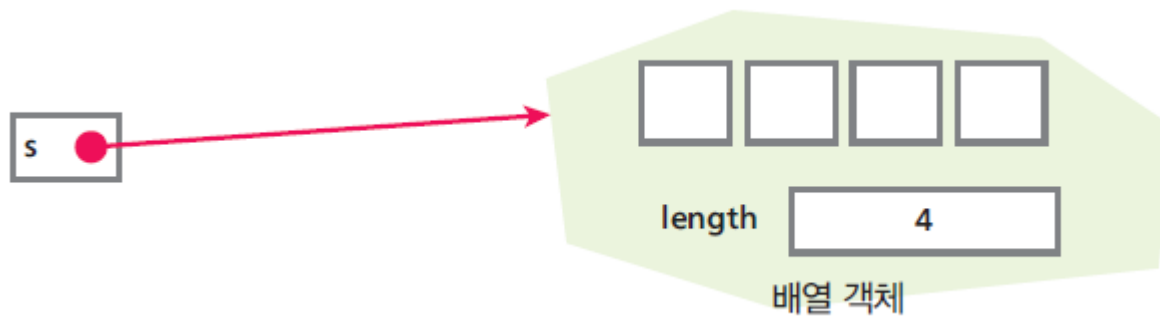
```
public class ArrayTest1 {  
  
    public static void main(String[] args) {  
  
        int[] s = new int[10];  
  
        for (int i = 0; i < s.length; i++) {  
            s[i] = i;  
        }  
  
        for (int i = 0; i < s.length; i++) {  
            System.out.print(s[i] + " ");  
        }  
    }  
}
```

0 1 2 3 4 5 6 7 8 9



배열의 크기

- 배열 이름이 s라면 s.length 가 배열의 크기가 된다.





배열의 초기화

```
public class ArrayTest3 {  
    public static void main(String[] args) {  
  
        int[] scores = { 10, 20, 30, 40, 50 };  
        for (int i = 0; i < scores.length; i++)  
            System.out.print(scores[i]+" ");  
    }  
}
```

10 20 30 40 50



for-each 루프

Syntax: for-each 문

```
int[] list = { 1, 2, 3, 4, 5 };  
for(int e : list) {  
    System.out.println(e);  
}
```

변수

배열



예제:

- 정수형 배열을 작성하고 **for-each** 루프로 배열에서 정수를 하나씩 꺼내서 화면에 출력하여 보자.

```
public class ArrayTest4 {  
    public static void main(String[] args) {  
        int[] numbers = { 10, 20, 30 };  
        for (int value : numbers)  
            System.out.print(value+" ");  
    }  
}
```

10 20 30



예제: 문자열 배열

- 앞에서는 정수 배열만을 살펴보았는데 실수 배열이나 문자열의 배열도 얼마든지 생성하여 사용할 수 있다. 여기서는 **5**가지의 피자 토핑의 종류를 문자열 배열에 저장하고 배열에 저장된 문자열을 꺼내서 화면에 출력하여 보자. **for-each** 루프를 사용해보자.

Pepperoni Mushrooms Onions Sausage Bacon



예제: 문자열 배열

```
public class PizzaTopping {  
    public static void main(String[] args) {  
  
        String[] toppings = { "Pepperoni", "Mushrooms", "Onions",  
"Sausage", "Bacon" };  
  
        for (String s: toppings) {  
            System.out.print(s + " ");  
        }  
    }  
}
```

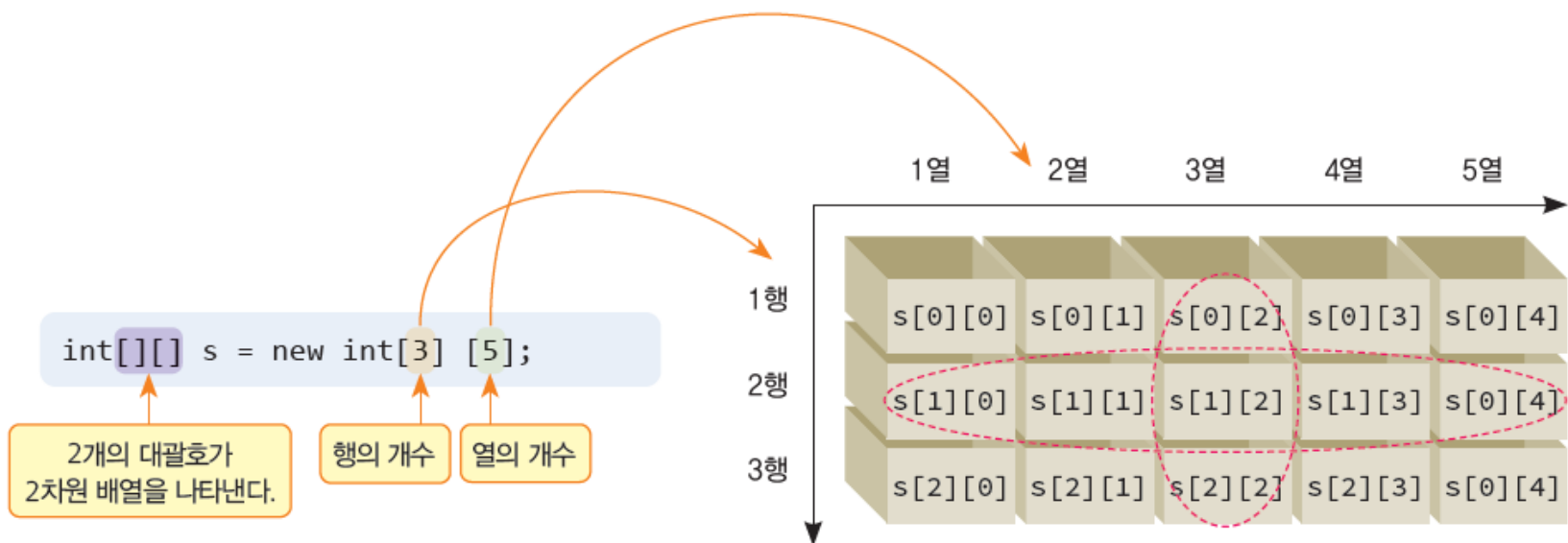



정가점거 중간문제

1. 10개의 부동소수점 난수를 저장하는 배열 faarray를 선언하고 생성하는 코드를 작성하라.
2. 배열의 크기는 어떻게 알 수 있는가?
3. 1번 문제의 farray를 for-each 구문으로 방문해서 출력하는 코드를 작성하라.
4. 1번 문제의 배열 farray에 저장된 값들의 합계를 계산하여 출력하는 코드를 작성하라.



2차원 배열





2차원 배열과 중첩된 루프

```
for(int i=0; i <3; i++)  
    for(int j=0; j<5; i++)  
        System.out.println(s[i][j]);
```

```
int[][] testArray = {  
    {10, 20, 30},  
    {40, 50, 60},  
    {70, 80, 90}  
};
```

2차원 배열의 초기화



예제: 극장 관객 수 계산

- 극장에 앉아있는 관객들을 2차원 배열로 나타낼 수 있다. 관객이 있는 좌석은 1로, 관객이 없는 좌석은 0으로 나타낸다. 현재 극장에 앉아있는 관객들의 수를 세는 프로그램을 작성해보자.

현재 관객 수는 7명입니다.



예제: 극장 관객 수 계산

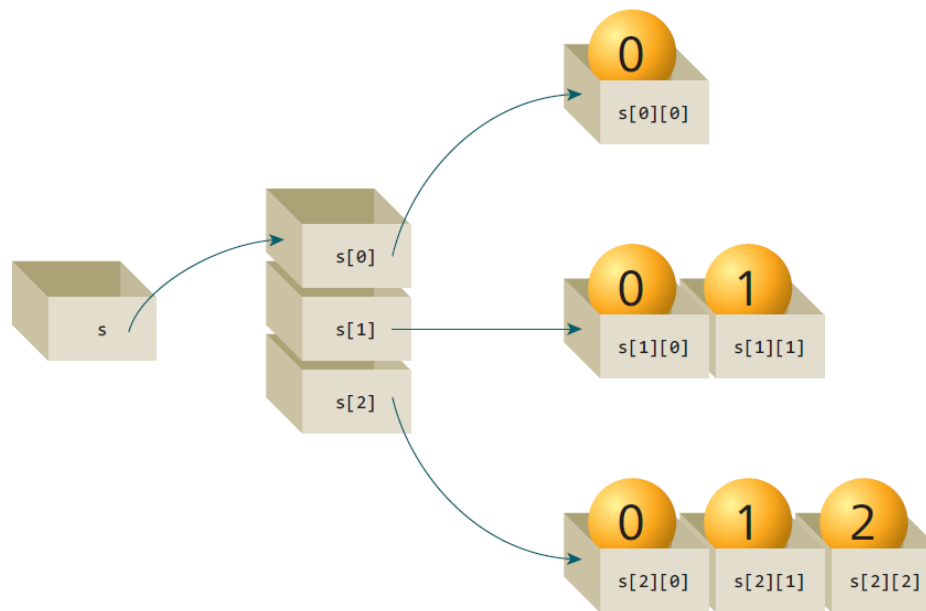
```
public class TheaterSeats {  
    public static void main(String[] args) {  
  
        int [][] seats = { {0, 0, 0, 1, 1, 0, 0, 0, 0, 0},  
                           {0, 0, 1, 1, 0, 0, 0, 0, 0, 0},  
                           {0, 0, 0, 0, 0, 0, 1, 1, 1, 0} };  
  
        int count=0;  
  
        for (int i = 0; i < seats.length; i++)  
            for (int k = 0; k < seats[i].length; k++)  
                count += seats[i][k];  
  
        System.out.print("현재 관객 수는 "+count+"명입니다.");  
    }  
}
```



래그드 배열

```
int[][] ragged = new int[MAX_ROWS+1][];
```

```
for(int r=0; r<=MAX_ROWS; r++)  
    ragged[r] = new int[r+1];
```





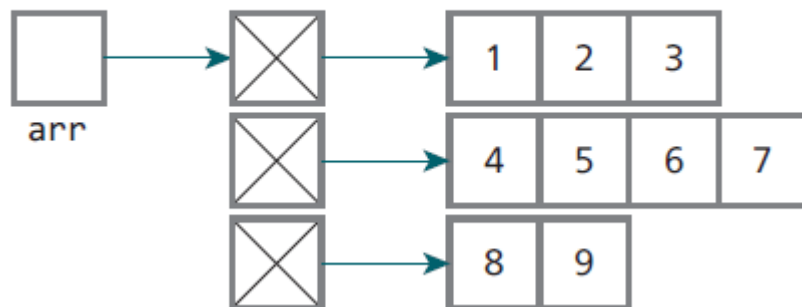
래그드 배열

```
public class RaggedArray {  
    public static void main(String[] args) {  
  
        int[][] ragged = new int[3][];  
        ragged[0] = new int[1];  
        ragged[1] = new int[2];  
        ragged[2] = new int[3];  
  
        for (int r = 0; r < ragged.length; r++)  
            for (int c = 0; c < ragged[r].length; c++)  
                ragged[r][c] = c;  
    }  
}
```



예제: 래그드 배열 생성

다음과 같은 래그드 배열을 생성해보자.





예제: 래그드 배열 생성

```
import java.util.Arrays;

public class RaggedArray2
{
    public static void main(String[] args)
    {
        int[][] rarray = new int[3][];

        rarray[0] = new int[] { 1, 2, 3, 4 };
        rarray[1] = new int[] { 5, 6, 7 };
        rarray[2] = new int[] { 8, 9 };

        for (int[] row: rarray) {
            System.out.println(Arrays.toString(row));
        }
    }
}
```

```
[1, 2, 3, 4]
[5, 6, 7]
[8, 9]
```



중간 점검

1. 다음의 배열 선언 중에서 잘못된 것을 모두 고르시오.

① `int[] a = int[10];`

② `int[] a = new int[10];`

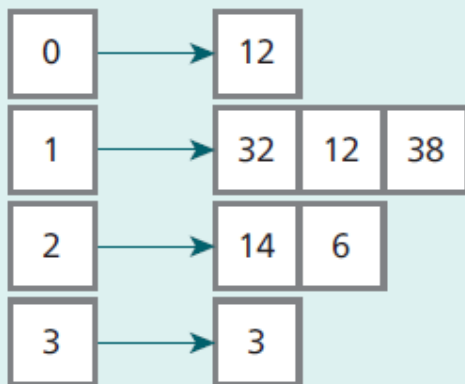
③ `int a[] = int[10];`

④ `int a[] = new int[10];`

⑤ `int[] a = { 1, 2, 3, 4, 5 };`

2. `double`형의 데이터를 저장하는 3×4 크기의 2차원 배열을 생성하라.

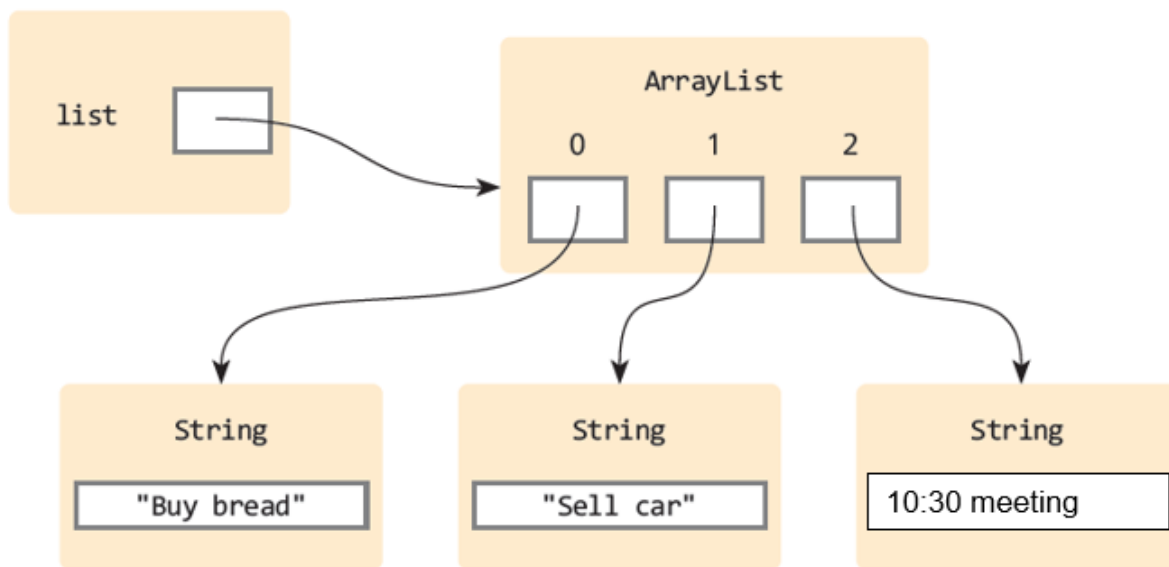
3. 초기값을 이용하여 다음과 같은 래그드 배열을 생성하는 코드를 작성하라.





ArrayList

- 자바에서는 **ArrayList** 라는 클래스를 제공하는데, 이 클래스를 사용하면 배열의 크기를 동적으로 변경하면서 사용할 수 있다.





ArrayList

ArrayList에 저장할 자료형

문자열을 저장하는
ArrayList 참조 변수 선언

Syntax: ArrayList 클래스

```
ArrayList<String> list;  
list = new ArrayList<>();
```

ArrayList 생성

```
list.add("Apple");  
list.add("Grape");
```



예제: 친구들의 리스트

```
import java.util.*;

public class ArrayListTest {
    public static void main(String args[]) {

        ArrayList<String> list = new ArrayList<>();
        list.add("철수");
        list.add("영희");
        list.add("순신");
        list.add("자영");
        for (String obj : list)
            System.out.print(obj + " ");

    }
}
```

철수 영희 순신 자영



Mini Project: 숫자 추측 게임

- 이 예제는 프로그램이 가지고 있는 정수를 사용자가 알아맞히는 게임이다. 사용자가 답을 제시하면 프로그램은 자신이 저장한 정수와 비교하여 제시된 정수가 더 높은지 낮은지 만을 알려준다



정답을 추측하여 보시오: 10
제시한 정수가 낮습니다.
정답을 추측하여 보시오: 30
제시한 정수가 낮습니다.
정답을 추측하여 보시오: 60
제시한 정수가 높습니다.
정답을 추측하여 보시오: 59
축하합니다. 시도횟수=4



Summary

- 조건문은 조건에 따라 여러 실행 경로로 분기하는 구조로서 **if-else** 문, **switch** 문이 있다.
- **break**문은 현재의 반복을 중단하고 반복 루프를 빠져나갈 때 사용한다.
- **switch** 문의 **case** 절에는 정수, 문자, 문자열이 올 수 있다.
- 반복문은 조건에 따라 문장을 반복하여 실행하는 구조로서 **for** 문, **while** 문이 있다.
- **do-while** 문은 문장을 무조건 한번 실행한 후에 조건을 검사하여 반복한다.
- 반복문안에 다른 반복문을 둘 수 있다.
- 배열은 같은 자료형의 데이터를 여러 개 순차적으로 저장하는 자료 구조이다.
- 자바에서 배열은 객체이다.
- 자바에서 배열을 생성하려면 먼저 참조 변수부터 선언한다. 참조 변수는 배열 객체의 주소를 저장하는 변수이다. 이어서 배열을 생성하여 저장 공간을 확보한다.
- 정수 3개를 저장하는 배열은 `int[] a = new int[4];`와 같이 만든다.
- 배열의 크기는 배열 객체의 **length** 필드에 저장된다.





Q & A

