A switch-case structure takes actions depending on the target variable.

```java
... switch (target) {
  case v1:
    // statements
    break;
  case v2:
    ...
    ...
  case vk:
    // statements
    break;
  default:
    // statements
}
...
A switch-case statement is more convenient than an if statement for multiple discrete conditions.

The variable *target*, always enclosed in parentheses, must yield a value of *char*, *byte*, *short*, *int*, or *String* type.

The value $v_1, \ldots, v_k$ must have the same data type as the variable *target*.

In each case, a *break* statement is a must.\(^1\)

- *break* is used to break a construct!

The default case, which is optional, can be used to perform actions when none of the specified cases matches *target*.

- Counterpart to *else* statements.

---

\(^1\)If not, there will be a fall-through behavior.
Example

```java
...  
    int trafficLight = (int) (Math.random() * 3);
    switch (trafficLight) {
        case 0:
            System.out.println("Red");  
            break;
        case 1:
            System.out.println("Yellow");
            break;
        case 2:
            System.out.println("Green");
            break;
    }
    ...
...  
```
A conditional expression evaluates an expression based on the specified condition and returns values accordingly.

```java
someVar = booleanExpr ? exprA : exprB;
```

- This is the only ternary operator in Java.
- If the `boolean` expression is evaluated `true`, then return `expr A`; otherwise, `expr B`. 
For example,

```
... 
if (num1 > num2)
    max = num1;
else
    max = num2;
...
```

Alternatively, one can use a conditional expression like this:

```
... 
max = (num1 > num2) ? num1 : num2;
... 
```
class Lecture4 {

    "Loops"

}

// keywords:
while, do, for, break, continue
A loop can be used to make a program execute statements repeatedly without having to code the same statements.

• For example, a program outputs “Hello, Java.” for 100 times.

```java
...  
System.out.println("Hello, Java.");
System.out.println("Hello, Java.");
.
.
// copy and paste for 100 times
.
System.out.println("Hello, Java.");
```

...
• This is a simple example to show the power of loops.
• In practice, any routine which repeats couples of times\(^2\) can be done by folding them into a loop.

\(^2\)I’d like to call them “patterns.”
Loops provide significant computational power.
Loops bring an efficient way of programming.
Loops could consume a lot of time.\(^3\)

\(^3\)We will visit the analysis of algorithms in the end of this lecture.
while Loops

A while loop executes statements repeatedly while the condition is true.

```
...  
while (condition) {
    // loop body
}
...  
```

- The condition should be a Boolean expression which determines whether or not the execution of the body occurs.
- If true, the loop body is executed and check the condition again.
- Otherwise, the entire loop terminates.
Example

Write a program which sums up all integers from 1 to 100.

- In math, the question can be written as:

  \[ \text{sum} = 1 + 2 + \cdots + 100. \]

- But this form is not doable in the machine.\(^4\)

---

\(^4\)We need to develop computational thinking. Read http://rsta.royalsocietypublishing.org/content/366/1881/3717.full or http://blog.orangeapple.tw/posts/what-is-computational-thinking/.
• Normally, the machine executes the instructions **sequentially**.

• So one needs to decompose the math equation into several steps, like:

```
...  
int sum = 0;
sum = sum + 1;
sum = sum + 2;
... 
sum = sum + 100;
...
```

• It is obvious that many similar statements can be found.
Using a while loop, the program can be rearranged as follows:

```java
int sum = 0;
int i = 1;
while (i <= 100) {
    sum = sum + i;
    ++i;
}
```

You should guarantee that the loop will terminate as expected. In practice, the number of loop steps (iterations) is unknown until the input data is given.
Malfunctioned Loops

- It is really easy to make an **infinite loop**.

```java
... 
while (true);
...
```

- The common errors of the loops are:
  - never start
  - never stop
  - not complete
  - exceed the expected number of iterations
Example

Write a program which asks the sum of two random integers and lets the user repeatedly enter a new answer until correct.

```java
Scanner input = new Scanner(System.in);
int x = (int) (Math.random() * 10);
int y = (int) (Math.random() * 10);
int ans = x + y;

System.out.println(x + " + " + y + " = ? ");
int z = input.nextInt();

while (z != ans) {
    System.out.println("Try again? ");
    z = input.nextInt();
}
System.out.println("Correct.");
input.close();
```
Loop Design Strategy

- Writing a correct loop is not an easy task for novice programmers.
- Consider 3 steps when writing a loop:
  - **Find the pattern**: identify the statements that need to be repeated.
  - **Wrap by loops**: put these statements in the loop.
  - **Set the continuation condition**: translate the criterion from the real world problem into computational conditions.\(^5\)

\(^5\)Not unique.
Another common technique for controlling a loop is to designate a special value when reading and processing a set of values.

- This special input value, known as a **sentinel value**, signifies the end of the loop.
- For example, the operating systems and the GUI apps.
Example: Cashier Problem

Write a program which sums positive integers from the input except for -1 to exit, and displays the sum.

```java
Scanner input = new Scanner(System.in);
System.out.println("Enter price (-1 to exit): ");
int currItem = input.nextInt();
int sum = 0;
while (currItem != -1) {
    if (currItem > 0) sum += currItem;
    System.out.println("Enter price (-1 to exit): ");
    currItem = input.nextInt();
}
System.out.println("Sum = " + sum);
input.close();
```

- Line 8 and 9 are the recurrence of Line 3 and 4?!
A do-while loop is similar to a while loop except that it does execute the loop body first and then checks the loop continuation condition.

```java
... do {
    // loop body
} while (condition); // Do not miss the semicolon!
...
```

- Note that there is a semicolon at the end of the do-while loop.
- The do-while loops are also called posttest loops, in contrast to while loops, which are pretest loops.
Write a program which sums positive integers from consecutive inputs and displays the sum when the input is \(-1\).

```java
... Scanner input = new Scanner(System.in);
int sum = 0;
int currItem;
do {
    System.out.println("Enter price (-1 to exit): ");
    currItem = input.nextInt();
    if (currItem >= 0) sum += currItem;
} while (currItem != -1);
System.out.println("Sum = "+sum);
input.close();
...
Exercise

Write a program which allows the user to enter positive integers consecutively and outputs the maximum when the input is $-1$.

? (-1 to exit)
10
? (-1 to exit)
20
? (-1 to exit)
-10
? (-1 to exit)
-1
Max = 20
Scanner input = new Scanner(System.in);
int max;

do {
    System.out.println("x? (−1 to exit) ");
    int x = input.nextInt();
    if (max < x) {
        max = x;
    }
} while (x != −1);

input.close();
A for loop generally uses a variable to control how many times the loop body is executed and when the loop terminates.

```
... for (init_action; condition; increment) {
    // loop body
}
... 
```

- *init-action*: declare and initialize a variable
- *condition*: set a criterion for loop continuation
- *increment*: how the variable changes after each iteration
- Note that these three terms are separated by semicolons.
Example

Sum from 1 to 100

Write a program which sums from 1 to 100.

```java
... int sum = 0;
for (int i = 1; i <= 100; ++i)
    sum = sum + i;
...
```

- Compared to the `while` version,

```java
... int sum = 0;
int i = 1;
while (i <= 100) {
    sum = sum + i;
    ++i;
}
...
```
Example: Selection Resided in Loop

Display all even numbers

Write a program which displays all even numbers smaller than 100.

• An even number is an integer of the form $x = 2k$, where $k$ is an integer.
• You may use the modular operator (%).

```java
... for (int i = 1; i <= 100; i++) {
    if (i % 2 == 0) System.out.println(i);
}
...```

• Also consider this alternative:

```java
... for (int i = 2; i <= 100; i += 2) {
    System.out.println(i);
}
...```

• How about odd numbers?
Example: Monte Carlo Simulation

- Write a program which conducts a Monte Carlo simulation to estimate $\pi$.

---

Jump Statements

The keyword **break** and **continue** are often used in repetition structures to provide additional controls.

- **break**: the loop is terminated right after a **break** statement is executed.
- **continue**: the loop skips this iteration right after a **continue** statement is executed.
- In practice, jump statements in loops should be conditioned.
Example: Primality

Write a program which determines if the input integer is a prime number.

- Let \( x > 1 \) be any natural number.
- Then \( x \) is said to be a prime number if \( x \) has no positive divisors other than 1 and itself.
- It is then straightforward to check if it is prime by dividing \( x \) by all natural numbers smaller than \( x \).
- For speedup, you can divide \( x \) by only numbers smaller than \( \sqrt{x} \). (Why?)
Scanner input = new Scanner(System.in);
System.out.println("Enter x > 2?");
int x = input.nextInt();
boolean isPrime = true;
input.close();

for (int y = 2; y < Math.sqrt(x); y++) {
    if (x % y == 0) {
        isPrime = false;
        break;
    }
}

if (isPrime) {
    System.out.println("Prime");
} else {
    System.out.println("Composite");
}
Exercise (Revisited)

- Redo the cashier problem by using an infinite loop with a break statement.

```java
... while (true) {
    System.out.println("Enter price (-1 to exit): ");
    currItem = input.nextInt();
    if (currItem >= 0) {
        amount += currItem;
    } else if (currItem == -1) {
        System.out.println("Total = " + amount);
        break;
    }
} ...
```
Another Example

Write a program which determines the holding years for an investment doubling its value.

- Let $curr$ be the current amount, $goal$ be the goal of this investment, and $r$ be the annual interest rate.
- Then this investment should take at least $n$ years so that the balance of the investment can double its value.
- Recall that the compounding formula is given by

$$curr = curr \times (1 + r/100).$$
... int r = 18; // 18%
int curr = 100;
int goal = 200;

int years = 0;
while (curr <= goal) {
    curr *= (1 + r / 100.0);
    years++;
}
System.out.println("Years = " + years);
System.out.println("Balance = " + curr);
...
A for loop can be an infinite loop by setting true or simply leaving empty in the condition statement.

An infinite for loop with an if-break statement is equivalent to a normal while loop.
In general, a **for** loop may be used if the number of repetitions is known in advance. If not, a **while** loop is preferred.