Control Constructs: Selection Statements

- Selection statements are used to perform different actions based on specific conditions. They help in decision-making in programming.

1. If Statement

- The `if` statement allows you to execute a block of code if a specified condition is true.

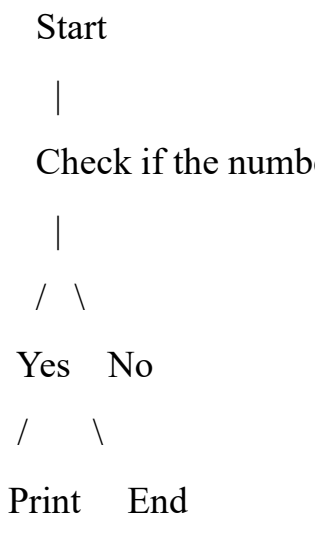
Syntax:

```
if (condition) {
   // code to be executed if a condition is true
}
```

Example:

```
int number = 10;
if (number > 0) {
   System.out.println("The number is positive.");
}
```

Flowchart:



2. If-Else Statement

- The `if-else` statement provides an alternative path of execution if the condition is false.

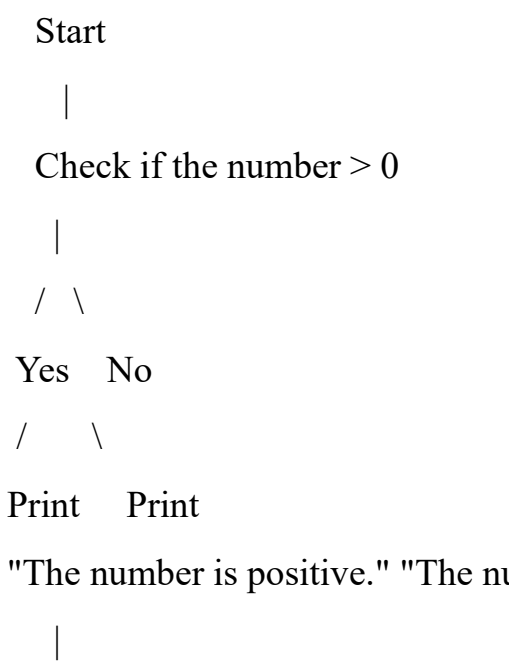
Syntax:

```
if (condition) {
   // code if a condition is true
} else {
   // code if a condition is false
}
```

Example:

```
int number = -5;
if (number > 0) {
   System.out.println("The number is positive.");
} else {
   System.out.println("The number is negative or zero.");
}
```

Flowchart:



End

3. Else If Ladder

- The `else if` ladder allows multiple conditions to be checked sequentially. It helps to test various conditions without nesting too many `if` statements.

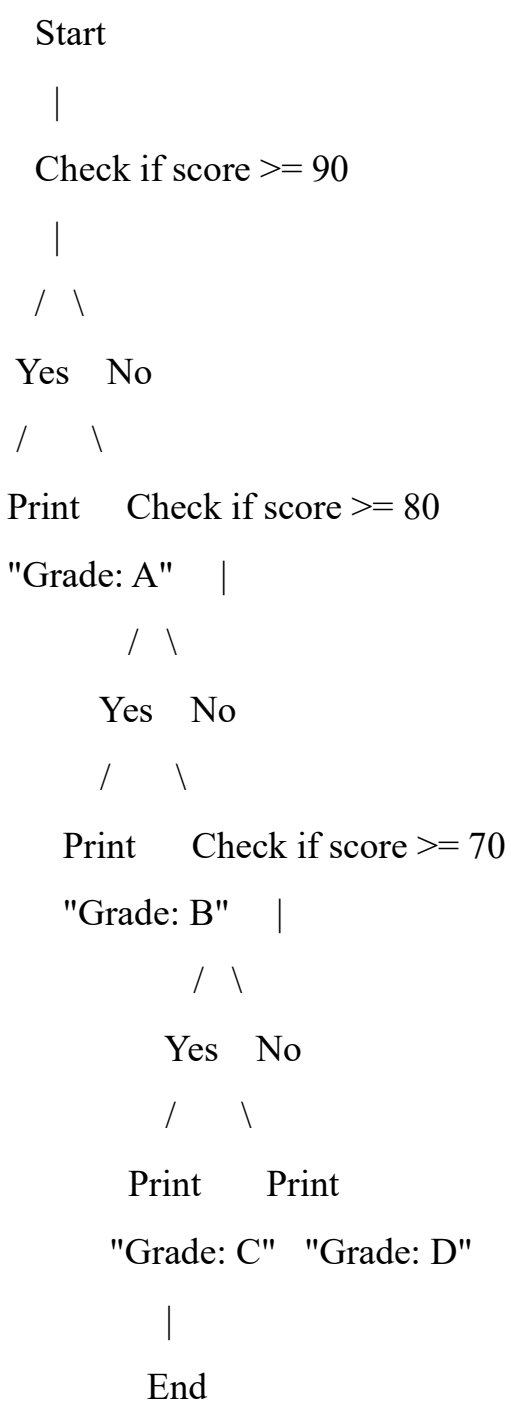
Syntax:

```
if (condition1) {
   // code if condition1 is true
} else if (condition2) {
   // code if condition2 is true
} else {
   // code if all conditions are false
}
```

Example:

```
int score = 75;
if (score >= 90) {
   System.out.println("Grade: A");
} else if (score >= 80) {
   System.out.println("Grade: B");
} else if (score >= 70) {
   System.out.println("Grade: C");
} else {
   System.out.println("Grade: D");
}
```

Flowchart:


Start
Check if score >= 90
Yes
No
Print "Grade: A"
Check if score >= 80
Yes
No
Print "Grade: B"
Check if score >= 70
Yes
No
Print "Grade: C"
Print "Grade: D"
End

4. Switch Statement

- The `switch` statement is a control statement that allows a variable to be tested for equality against a list of values. Each value is called a case.

Syntax:

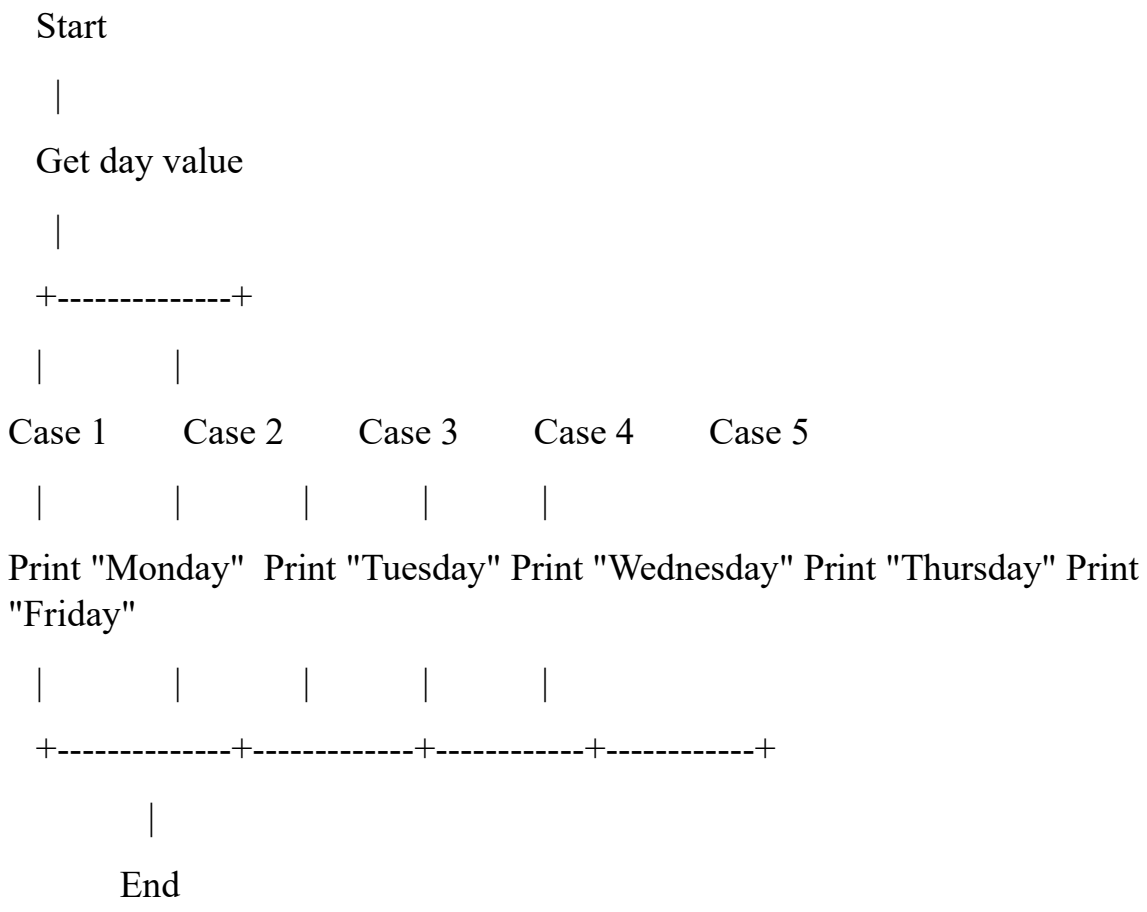
```
switch (variable) {
   case value1:
      // code for value1
      break;
   case value2:
      // code for value2
      break;
   default:
      // code if no case matches
}
```

Example:

```
int day = 4;
switch (day) {
   case 1:
      System.out.println("Monday");
      break;
   case 2:
      System.out.println("Tuesday");
      break;
   case 3:
      System.out.println("Wednesday");
      break;
   case 4:
```

```
        System.out.println("Thursday");
        break;
    case 5:
        System.out.println("Friday");
        break;
    default:
        System.out.println("Invalid day");
}
```

Flowchart:



5. Nested If Statements

- You can nest `if` statements inside other `if` statements to check multiple conditions.

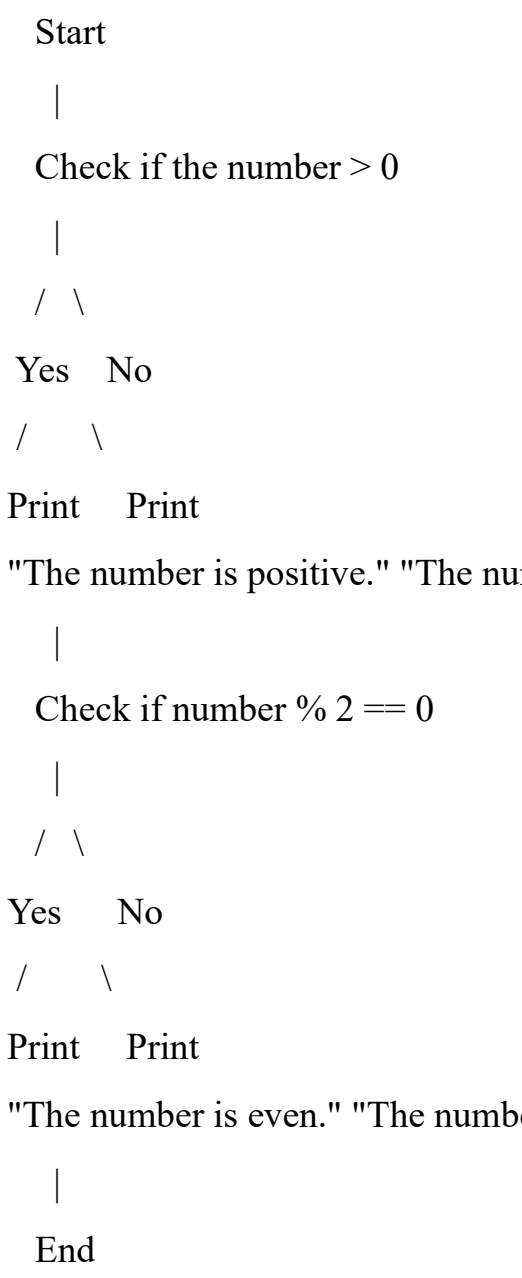
Syntax:

```
if (condition1) {
   if (condition2) {
      // code if condition1 and condition2 are true
   }
}
```

Example:

```
int number = 15;
if (number > 0) {
   System.out.println("The number is positive.");
   if (number % 2 == 0) {
      System.out.println("The number is even.");
   } else {
      System.out.println("The number is odd.");
   }
} else {
   System.out.println("The number is negative or zero.");
}
```

Flowchart:



Summary of Selection Statements

If Statement: Executes a block of code if a condition is true.

If-Else Statement: Provides a choice between two blocks of code based on a condition.

Else If Ladder: Tests multiple conditions in sequence.

Switch Statement: Chooses among multiple options based on the value of a variable.

Nested If Statements: Allows checking conditions within conditions for more complex decision-making.