Garfield AP CS

User Input, If/Else

Warmup

- Write a method add10 that takes one integer parameter. Your method should return the number passed in plus ten.

Input and `System.in`

- **interactive program**: Reads input from the console.

  - While the program runs, it asks the user to type input.
  - The input typed by the user is stored in variables in the code.

  - Can be tricky; users are unpredictable and misbehave.
  - But interactive programs have more interesting behavior.

- **`Scanner`**: An object that can read input from many sources.

  - Communicates with `System.in` (the opposite of `System.out`)
  - Can also read from files (Ch. 6), web sites, databases, …

Scanner syntax

- The `Scanner` class is found in the `java.util` package.

  ```
  import java.util.*;   // so you can use Scanner
  ```

- Constructing a `Scanner` object to read console input:

  ```
  Scanner name = new Scanner(System.in);
  ```

  – Example:

  ```
  Scanner console = new Scanner(System.in);
  ```

Scanner methods

Method	Description
nextInt()	reads an int from the user and returns it
nextDouble()	reads a double from the user
next()	reads a one-word String from the user
nextLine()	reads a one-*line* String from the user

- Each method waits until the user presses Enter.
- The value typed by the user is returned.

```
System.out.print("How old are you? ");  // prompt
int age = console.nextInt();
System.out.println("You typed " + age);
```

- **prompt**: A message telling the user what input to type.

Scanner example

```java
import java.util.*;    // so that I can use Scanner

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

        System.out.print("How old are you? ");
        int age = console.nextInt();

        int years = 65 - age;
        System.out.println(years + " years until retirement!");
    }
}
```

age 29

years 36

- Console (user input underlined):

```
How old are you? 29
36 years until retirement!
```

Scanner example 2

```java
import java.util.*;    // so that I can use Scanner

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

        System.out.print("Please type two numbers: ");
        int num1 = console.nextInt();
        int num2 = console.nextInt();

        int product = num1 * num2;
        System.out.println("The product is " + product);
    }
}
```

- Output (user input underlined):

```
Please type two numbers: 8 6
The product is 48
```

- The Scanner can read multiple values from one line.

Input tokens

- **token**: A unit of user input, as read by the `Scanner`.
  - Tokens are separated by *whitespace* (spaces, tabs, new lines).
  - How many tokens appear on the following line of input?
    ```
    23  John  Smith    42.0   "Hello world"
    $2.50   "   19"
    ```

- When a token is not the type you ask for, it crashes.
    ```
    System.out.print("What is your age? ");
    int age = console.nextInt();
    ```
    Output:
    ```
    What is your age? Timmy
    java.util.InputMismatchException
            at java.util.Scanner.next(Unknown Source)
            at java.util.Scanner.nextInt(Unknown Source)
            ...
    ```
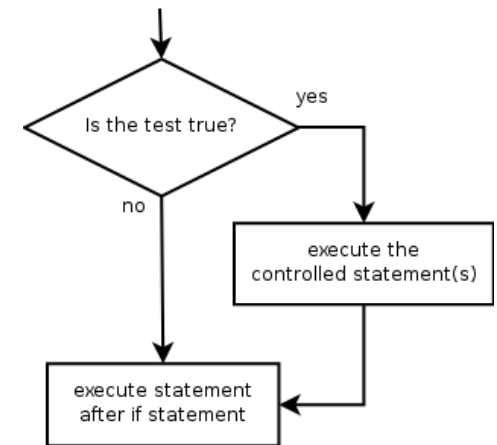
Conditionals

- Fundamental to creating complex behavior
- Especially useful when user input is involved
  - React differently based on what they entered

The `if` statement

Executes a block of statements only if a test is true

```
if (test) {
    statement;
    ...
    statement;
}
```
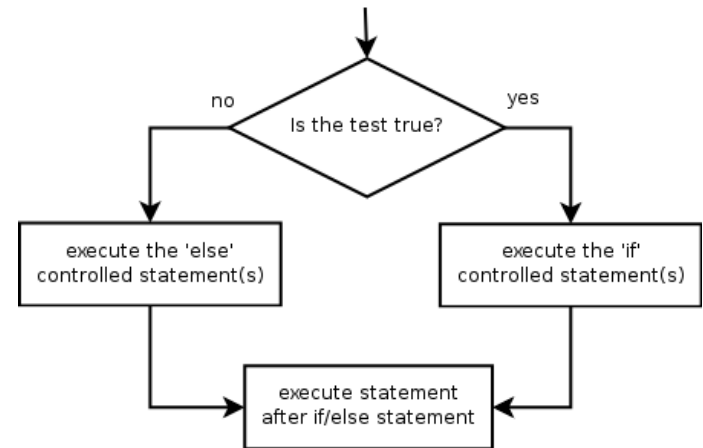

- Example:

```
double gpa = console.nextDouble();
if (gpa >= 2.0) {
    System.out.println("Application accepted.");
}
```

The `if/else` statement

Executes one block if a test is true, another if false

```
if (test) {
    statement(s);
} else {
    statement(s);
}
```


- Example:
```
double gpa = console.nextDouble();
if (gpa >= 2.0) {
    System.out.println("Welcome to Mars University!");
} else {
    System.out.println("Application denied.");
}
```

Relational expressions

- `if` statements and `for` loops both use logical tests.

  ```
  for (int i = 1; i <= 10; i++) { ...
  if (i <= 10) { ...
  ```

 - These are `boolean` expressions that use *relational operators*:

Operator	Meaning	Example	Value
==	equals	1 + 1 == 2	true
!=	does not equal	3.2 != 2.5	true
<	less than	10 < 5	false
>	greater than	10 > 5	true
<=	less than or equal to	126 <= 100	false
>=	greater than or equal to	5.0 >= 5.0	true

Misuse of `if`

- ## What's wrong with the following code?

```
Scanner console = new Scanner(System.in);
System.out.print("What percentage did you earn? ");
int percent = console.nextInt();
if (percent >= 90) {
    System.out.println("You got an A!");
}
if (percent >= 80) {
    System.out.println("You got a B!");
}
if (percent >= 70) {
    System.out.println("You got a C!");
}
if (percent >= 60) {
    System.out.println("You got a D!");
}
if (percent < 60) {
    System.out.println("You got an F!");
}
...
```
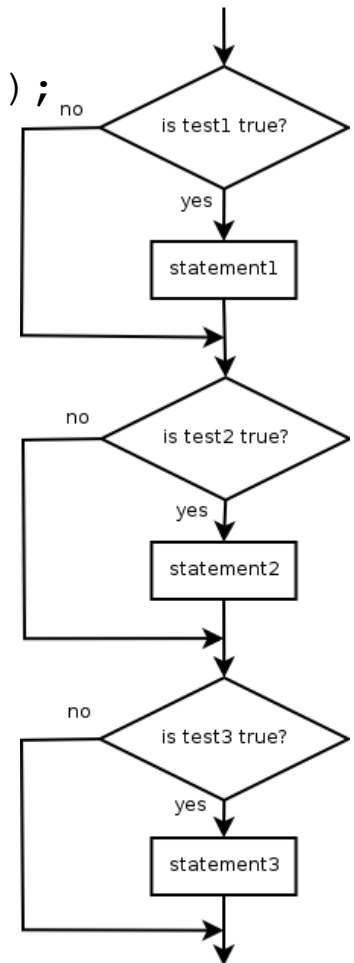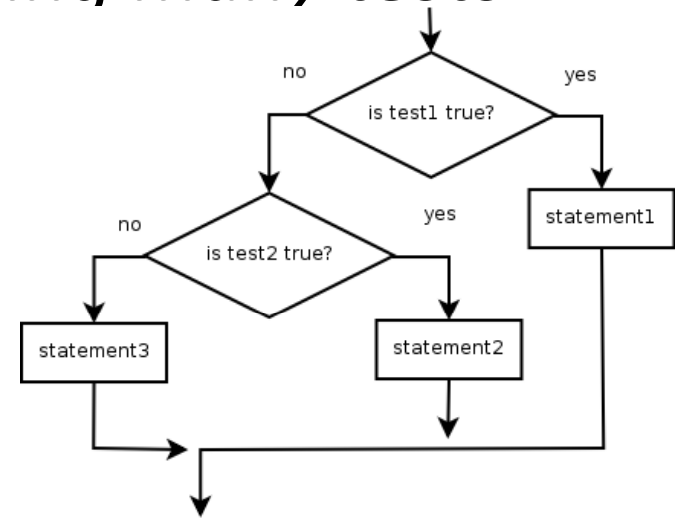
Nested `if/else`

Chooses between outcomes using many tests

```
if (test) {
    statement(s);
} else if (test) {
    statement(s);
} else {
    statement(s);
}
```


- Example:

```
if (x > 0) {
    System.out.println("Positive");
} else if (x < 0) {
    System.out.println("Negative");
} else {
    System.out.println("Zero");
}
```
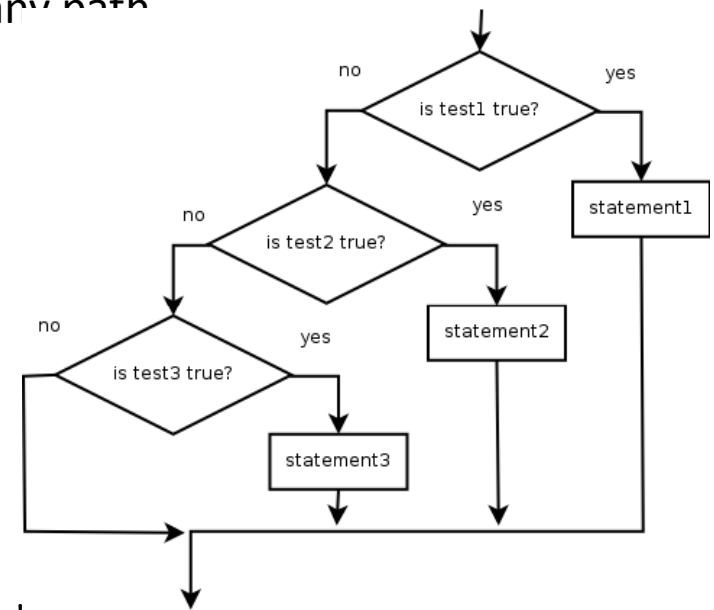
Nested `if/else/if`

– If it ends with `else`, exactly one path must be taken.
– If it ends with `if`, the code might not execute any path.

```
if (test) {
    statement(s);
} else if (test) {
    statement(s);
} else if (test) {
    statement(s);
}
```

- Example:

```
if (place == 1) {
    System.out.println("Gold medal!");
} else if (place == 2) {
    System.out.println("Silver medal!");
} else if (place == 3) {
    System.out.println("Bronze medal.");
}
```

Nested `if` structures

- exactly 1 path *(mutually exclusive)*

```
if (test) {
    statement(s);
} else if (test) {
    statement(s);
} else {
    statement(s);
}
```

- 0 or 1 path *(mutually exclusive)*

```
if (test) {
    statement(s);
} else if (test) {
    statement(s);
} else if (test) {
    statement(s);
}
```

- 0, 1, or many paths *(independent tests; not exclusive)*

```
if (test) {
    statement(s);
}
if (test) {
    statement(s);
}
if (test) {
    statement(s);
}
```

Which nested `if/else`?

- **(1) if/if/if   (2) nested if/else   (3) nested if/else/if**
  - Whether a user is lower, middle, or upper-class based on income.
    - **(2)**   nested `if / else if / else`

  - Whether you made the dean's list (GPA ≥ 3.8) or honor roll (3.5-3.8).
    - **(3)**   nested `if / else if`

  - Whether a number is divisible by 2, 3, and/or 5.
    - **(1)**   sequential `if / if / if`

  - Computing a grade of A, B, C, D, or F based on a percentage.
    - **(2)**   nested `if / else if / else if / else if / else`