

COMP 110/L Lecture 12

Maryam Jalali

Some slides adapted from Dr. Kyle Dewey

Outline

O switch

switch

Problem

`if` is verbose when checking many conditions.

Problem

`if` is verbose when checking many conditions.

```
if (x == 5) {  
    return "foo";  
} else if (x == 6) {  
    return "bar";  
} else if (x == 7) {  
    return "baz";  
} else if (x == 8) {  
    return "blah";  
} else {  
    return "unknown";  
}
```

switch

`switch` allows for multiple `==` conditions to be checked

```
if (x == 5) {  
    return "foo";  
} else if (x == 6) {  
    return "bar";  
} else if (x == 7) {  
    return "baz";  
} else if (x == 8) {  
    return "blah";  
} else {  
    return "unknown";  
}
```

switch

switch allows for multiple == conditions to be checked

```
if (x == 5) {  
    return "foo";  
} else if (x == 6) {  
    return "bar";  
} else if (x == 7) {  
    return "baz";  
} else if (x == 8) {  
    return "blah";  
} else {  
    return "unknown";  
}
```

```
switch (x) {  
case 5:  
    return "foo";  
case 6:  
    return "bar";  
case 7:  
    return "baz";  
case 8:  
    return "blah";  
default:  
    return "unknown";  
}
```

Example:

`SwitchBasic.java`

switch Semantics

- Look at the thing you're `switching` on
- Jump to the applicable case
- Keep running statements until something **stops** you

switch Semantics

- Look at the thing you're `switching` on
- Jump to the applicable case
- Keep running statements until something stops you

```
switch (x) {  
  case 1:  
    return "hi";  
  case 2:  
    System.out.println("bye");  
  default:  
    System.out.println("huh");  
}
```

switch Semantics

- Look at the thing you're `switching` on
- Jump to the applicable case
- Keep running statements until something stops you

```
switch (1) {  
  case 1:  
    return "hi";  
  case 2:  
    System.out.println("bye");  
  default:  
    System.out.println("huh");  
}
```

switch Semantics

- Look at the thing you're `switching` on
- Jump to the applicable case
- Keep running statements until something stops you

```
switch (1) {  
→ case 1:  
    return "hi";  
case 2:  
    System.out.println("bye");  
default:  
    System.out.println("huh");  
}
```

switch Semantics

- Look at the thing you're `switching` on
- Jump to the applicable case
- Keep running statements until something stops you

```
switch (1) {  
  case 1:  
    → return "hi";  
  case 2:  
    System.out.println("bye");  
  default:  
    System.out.println("huh");  
}
```

switch Semantics

- Look at the thing you're `switching` on
- Jump to the applicable case
- Keep running statements until something stops you

```
switch (1) {  
  case 1:  
    → return "hi";  
  case 2:  
    System.out.println("bye");  
  default:  
    System.out.println("huh");  
}
```

switch Semantics

- Look at the thing you're `switching` on
- Jump to the applicable case
- Keep running statements until something stops you

```
switch (3) {  
  case 1:  
    return "hi";  
  case 2:  
    System.out.println("bye");  
  default:  
    System.out.println("huh");  
}
```

switch Semantics

- Look at the thing you're `switching` on
- Jump to the applicable case
- Keep running statements until something stops you

```
switch (3) {  
  case 1:  
    return "hi";  
  case 2:  
    System.out.println("bye");  
  → default:  
    System.out.println("huh");  
}
```


switch Semantics


- Look at the thing you're `switching` on
- Jump to the applicable case
- Keep running statements until something stops you

```
switch (3) {  
  case 1:  
    return "hi";  
  case 2:  
    System.out.println("bye");  
  default:  
    → System.out.println("huh");  
}
```

switch Semantics

- Look at the thing you're `switching` on
- Jump to the applicable case
- Keep running statements until something stops you

```
switch (3) {  
  case 1:  
    return "hi";  
  case 2:  
    System.out.println("bye");  
  default:  
    System.out.println("huh");  
}
```



switch Semantics

- Look at the thing you're `switching` on
- Jump to the applicable case
- Keep running statements until something stops you

```
switch (2) {  
  case 1:  
    return "hi";  
  case 2:  
    System.out.println("bye");  
  default:  
    System.out.println("huh");  
}
```

switch Semantics

- Look at the thing you're `switching` on
- Jump to the applicable case
- Keep running statements until something stops you

```
switch (2) {  
  case 1:  
    return "hi";  
  → case 2:  
    System.out.println("bye");  
  default:  
    System.out.println("huh");  
}
```

switch Semantics

- Look at the thing you're `switching` on
- Jump to the applicable case
- Keep running statements until something stops you

```
switch (2) {  
  case 1:  
    return "hi";  
  case 2:  
    → System.out.println("bye");  
  default:  
    System.out.println("huh");  
}
```

switch Semantics

- Look at the thing you're `switching` on
- Jump to the applicable case
- Keep running statements until something stops you

```
switch (2) {  
  case 1:  
    return "hi";  
  case 2:  
    System.out.println("bye");  
  default:  
    → System.out.println("huh");  
}
```

switch Semantics

- Look at the thing you're `switching` on
- Jump to the applicable case
- Keep running statements until something stops you

```
switch (2) {  
  case 1:  
    return "hi";  
  case 2:  
    System.out.println("bye");  
  default:  
    System.out.println("huh");  
  → }  
}
```

Example:

```
SwitchFallthrough.java
```


Preventing “fall-through”

The `break` statement will exit out of a `switch`.

Preventing “fall-through”

The `break` statement will exit out of a `switch`.

```
switch (x) {  
  case 1:  
    return "hi";  
  case 2:  
    System.out.println("bye");  
  default:  
    System.out.println("huh");  
}
```

Preventing “fall-through”

The `break` statement will exit out of a `switch`.

```
switch (x) {  
  case 1:  
    return "hi";  
  case 2:  
    System.out.println("bye");  
    break;  
  default:  
    System.out.println("huh");  
}
```

Preventing “fall-through”

The `break` statement will exit out of a `switch`.

```
switch (2) {  
  case 1:  
    return "hi";  
  case 2:  
    System.out.println("bye");  
    break;  
  default:  
    System.out.println("huh");  
}
```

Preventing “fall-through”

The `break` statement will exit out of a `switch`.

```
switch (2) {  
  case 1:  
    return "hi";  
  → case 2:  
    System.out.println("bye");  
    break;  
  default:  
    System.out.println("huh");  
}
```

Preventing “fall-through”

The `break` statement will exit out of a `switch`.

```
switch (2) {  
  case 1:  
    return "hi";  
  case 2:  
    → System.out.println("bye");  
    break;  
  default:  
    System.out.println("huh");  
}
```

Preventing “fall-through”

The `break` statement will exit out of a `switch`.

```
switch (2) {  
  case 1:  
    return "hi";  
  case 2:  
    System.out.println("bye");  
→ break;  
  default:  
    System.out.println("huh");  
}
```

Preventing “fall-through”

The `break` statement will exit out of a `switch`.

```
switch (2) {  
  case 1:  
    return "hi";  
  case 2:  
    System.out.println("bye");  
    break;  
  default:  
    System.out.println("huh");  
  }  
→
```


Preventing “fall-through”

The `break` statement will exit out of a `switch`.

```
int roll = 3 ;  
switch ( roll )  
{  
  case 1 :  
    printf("I am Pankaj");  
    break;  
  case 2 :  
    printf("I am Nikhil");  
    break;  
  case 3 :  
    printf("I am John");  
    break;  
  default :  
    printf("No student found");  
    break;  
}
```

Example:

`SwitchBreak.java`

Some Important rules for switch statements:

- Duplicate case values are not allowed.
- The value for a case must be the same data type as the variable in the switch.
- The value for a case must be a constant. Variables are not allowed.
- The break statement is used inside the switch to terminate a statement sequence.
- The break statement is optional. If omitted, execution will continue on into the next case.
- A switch works with *int* and ***String***.

```
snum = user_input.nextDouble();

ans = fnum - snum;
System.out.println("Answer is: " + ans);
break;

case 3:
    System.out.println("You choose Multiplication");
    System.out.print("Enter first num: ");
    fnum = user_input.nextDouble();

    System.out.print("Enter second num: ");
    snum = user_input.nextDouble();

    ans = fnum * snum;
    System.out.println("Answer is: " + ans);
break;

case 4:
    System.out.println("You choose Division");
    System.out.print("Enter first num: ");
    fnum = user_input.nextDouble();

    System.out.print("Enter second num: ");
    snum = user_input.nextDouble();

    ans = fnum / snum;
    System.out.println("Answer is: " + ans);
break;

default:
    System.out.println("You can choose from number 1 to 4 only");
break;
}
}
}
```

switch and Testing

Each case is a test candidate, as is default.

switch and Testing

Each case is a test candidate, as is default.

```
int result = 0;
switch (input) {
case 1:
    result = result + 2;
case 2:
    result = result + 5;
default:
    result = result + 12;
}
```

switch and Testing

Each case is a test candidate, as is default.

```
1  int result = 0;
   switch (input) {
   case 1:
       result = result + 2;
   case 2:
       result = result + 5;
   default:
       result = result + 12;
   }
```

switch and Testing

Each case is a test candidate, as is default.

```
int result = 0;
switch (input) {
1 case 1:
    result = result + 2;
2 case 2:
    result = result + 5;
default:
    result = result + 12;
}
```


switch and Testing

Each case is a test candidate, as is default.

```
int result = 0;
switch (input) {
1 case 1:
    result = result + 2;
2 case 2:
    result = result + 5;
3 default:
    result = result + 12;
}
```