Switches in Java

- The switch statement is a more compact expressions certain types of linear nested if-else.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

- Keyword: Switch,

- variable: quizGrade

- The variable quizGrade was being used as a comparison for each condition of the if-else statement (the first picture).

- For the switch, we will have different cases that will then be applied to an operation. When the operation is finished and we want to start a new case, we will use the key word break;

That’s going to drop us out of the switch statement to the new operations for further cases. It’s kind of like when we’ve finished an if statement and then we go to the next else or if-else.

General switch statement:

Switch (expression)

{

Case value1:

// code to do if expression equals value 1

Break;

Case value2:

// code to do if expression equals value 2

Break;

. . .

Default:

// code to do if no other cases are matched

- Sometimes we do leave out the break. It basically results in an or statement.

- PRO-TIP: Ask another programmer for their opinion.

- Switch statements only work on certain data types: int, char, String, short, and byte.

Can we use wrappers?? NO.

- Switches (in some cases) are useful as opposed to nested if-else statements because:

- It can be more compact. Less typing.

- Easier to read.

- A little more efficient in execution.

If (classAverage >= 90 && examAverage >= 85)

{

System.out.println(letterGrade = ‘A’);

{

Else if ( classAverage >= 85 && examAverage >= 75)

{

System.out.println(letterGrade = ‘B’);

}

…

Double cost = 200

if ( length \* width >= 750)

{

Cost += 50;

{

If ( woodCode == 1 );

{

Cost = cost + 100

{

Else if (woodCode == 2 );

{

S.o.pln (cost);