Sample code above

Bigger done

New string is BIGGER

Output:

System.out.println("done");
System.out.println("33");
System.out.print("33");
// print and stay on same line
System.out.println("new string is ", + 33);
// print and skip to newline
String s3 = s2, toulper( case() );
String s2 = s1, replace ( "f", "g" );
String s1 = "blitter";

public static void main (String[] args) {

public class convertString { save as convertString.java;

To: convert blitter to bigger

May be empty:
parameters/arguments/input

May be void:
return

Return type:
method name
Strings try to mimic primitive data types.

Most other classes require an import from Java library.

`Scanner` object - reads user input

```java
import java.util.Scanner

public class Input {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        System.out.println("Enter day ");

        String day = scan.nextLine();

        System.out.println("Happy " + day);
        scan.close();
    }
}
```

(Just as `int x = 5`)

(type name of value)
What to use to input `date` first? `nextInt()`

Let's try (see `Sample Code input.java`):

```java
int date = scan.nextInt();
...
String day = scan.nextLine();
```

What went wrong?

User's input:

- `14
  date`
- `Friday` → not scanned

How to fix?

⇒ Use another `nextLine()` after `nextInt()`

```java
int date = scan.nextInt();
scan.nextLine(); // read + ignoring "\n"
String day = scan.nextLine();
```
Different data types lead to need to convert data from one type to another.

Cast: - convert the data type of a variable

double x = 5.0;
int y;

y = x;  // won't compile.

y = (int) x;  // casting y = 5, required because we are losing info  
              // 5.0 turned into 5

What happens here?

x = 5.75;
y = (int) x;  // rounds down, y = 5

What about:

int z = 7;
double w = z;  // compiles, casting not needed  
                // because we are "gaining"  
                // information (turning 7 into 7.0).
Conditionals (if/else)

Comparative operators: < > <= >= == !=
Logical: " || (or) && (and) ! (not)

if (condition) {} // if - block

// body
/* Must use && if body contains (declaration or)
more than 1 statement */

else-if (condition) // else-if block
    // can have multiple else-if blocks
else // else-block (no condition)
    // body.

Some examples:

1) Input an integer, set a boolean to true if integer is positive

   // Sample Code: ifelse.java
boolean valid;
Scanner scan = new Scanner(System.in); // requires import
int num = scan.nextInt();

if (num > 0)
    valid = true;
    System.out.println("valid");
else
    valid = false;

(2) Set valid to true if num < 20 or > 50

<<Sample Code (2 ways) if-else.java>>

if (num < 20)
    valid = true;
    System.out.println("valid");
else if (num > 50)
    valid = true;
    System.out.println("valid");
else
    valid = false;
2nd way -

```java
if (num < 20 || num > 50)
    valid = true;
else
    valid = false;
```

Blocks of code limit accessibility of variables.

Scope - area within a program where a variable can be accessed.

"Rule - variable can be accessed" - only within innermost 9 of its declaration (e.g. boolean valid).

ex:

```java
if (num < 20) { // 3 3 needed
    boolean valid2 = true; // valid2 declared
}
```

```java
System.out.print(valid2); // won't compile
```

Will it compile? No.

Where is compilation error?

How to fix? <Sample Code scope.java>
boolean valid2;

if (num <= 20)
    valid2 = true;
else
    valid2 = false;

System.out.println(valid2);  //compiles!

Loops

Looping made easier with increments:

i++ : i = i + 1
i-- : i = i - 1
i += 2 : i = i + 2
i -= k : i = i - k
i *= 5 : i = i * 5
3 Types of Loops:

(1) do-while 
  use when number of iterations 
  is not known in advance.

\[
\text{do-while:} \quad \begin{align*}
\text{do} \quad \text{while (condition)?} \\
\langle \text{loop body} \rangle \\
\text{while (condition):} \\
\end{align*}
\]

"Difference?" Do-loop executes at least once.

```
int count = 10; // not 0!
do {
    System.out.println(count);
    count ++;
} while (count < 5);
```

```
int count = 10; // not 0!
while (count < 5) {
    System.out.println(count);
    count ++;
}
```

"Output?" => 10 done
done

do-while: executes at least once
while: may not execute.
(3) for loop: use when number of iterations is known in advance.

Example: input 10 integers from user

```java
Scanner scan = new Scanner(System.in);
int num;
```

Break this up:

- **Initialization**: `i = 0`
- **Condition**: `i < 10`
- **Increment**: `i += 4`

```
System.out.println("Enter number:");
num = scan.nextInt();
```

Body (repeated)

Question: How to modify above to print sum of 5 integers?

```java
Scanner scan = new Scanner(System.in);
int sum = 0; // must initialize to 0!!
for (int i = 0; i < 5; i++) {
    System.out.println("Enter number:");
    num = scan.nextInt();
    sum += num;
}
```