```
    // instanceof => returns the type of a specified value
    // > < >= <= are also comparison operators
```

Control structures

```
// ---------- CONTROL STRUCTURE ----------

// CONDITIONALS

// IF ELSE IF ELSE

int age = 75;
if (age >= 75) {
    System.out.prinln("aight bet");
} else if (age == 18) {
    System.out.println("this rose is very nice");
} else {
    System.out.println("You are an adult!");
} // this prints out "aight bet" to the stdout

// SWITCH CASE BREAK DEFAULT
    // remember to include break statement to prevent logic from falling through all cases
    // default => specifies the default case if all other conditional checks fail

switch(Day) {
    case "Sunday": System.out.println("Today is Sunday!");
    break;
    case "Monday": System.out.println("Today is Monday!");
    break;
    case "Tuesday": System.out.println("Today is Tuesday!");
    break;
    default: System.out.println("Ohh shittt");
}

// TRY CATCH FINALLY
    // try => try block surrounds any potentially dangerous code, similar to try except in Python
    // catch => code within the catch block runs if try block fails, otherwise this catch block is ignored if the try block runs succesfully, and can receive specific error messages and variable names as required, there can also be multiple catch statements
    // finally => finally block always executes regardless of whether an exception is caught or not, placed at the end of a try catch chain

try {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter a whole number to divide: ");
    int x = scanner.nextInt();
    System.out.println("Enter a whole number to divide by: ");
    int y = scanner.nextInt();
    int z = x/y;
    System.out.println("result: " + z);
}

catch(ArithmeticExpression e) { // to handle the exception generated by the java error log in the stdout, "java.lang.ArithmeticExpression"
    System.out.println("Stop being clown please I beg");
}

catch(InputMismatchException e) { // to handle the exception generated by the java error log in the stdout, "java.lang.InputMismatchException"
    System.out.println("Enter a number you idiot!");
}

finally {
    System.out.println("This will alwyas print!");
    scanner.close();
}

// LOOPS

// WHILE LOOPS

while (true) {
    System.out.println("I'm trapped in an endless loop!");
} // a conventional while loop, though the while true has created an infinite loop

// DO WHILE LOOPS

String yes_or_no = "yes";
do {
    System.out.println("Eh ok");
} while (yes_or_no == "yes"); // also an infinite do while loop since no break condition set

// FOR LOOPS
    // allows for rudimentary implementation of a for loop like in other languages

for (int i=0; i <= 10; i++) {
    System.out.println(i);
} // this prints out 0 to 10 to the stdout, separated by newlines

// FOR EACH LOOP
    // : => specifies the iteration variable over an iterable data structure

String[] animals = {"cat", "dog", "bird", "elephant"};
for (String animal: animals) {
    System.out.println(animal); // this would iterate over every element in the String Array animals, assigning each element to the String variable animal, and printing said variable to the stdout
}
```