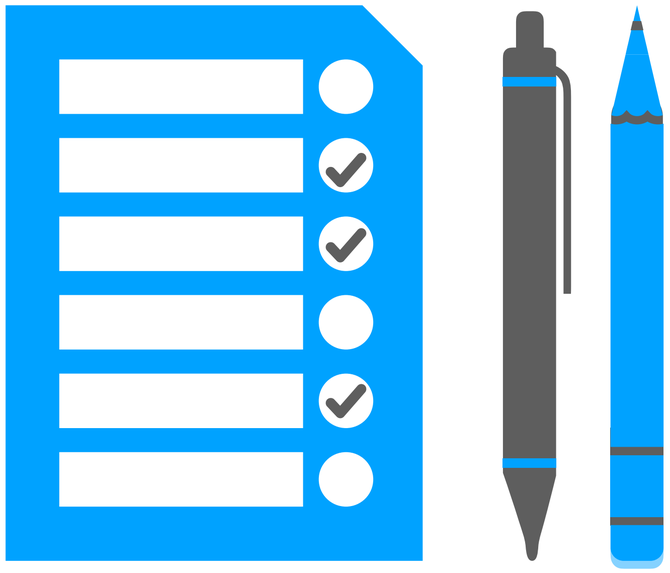
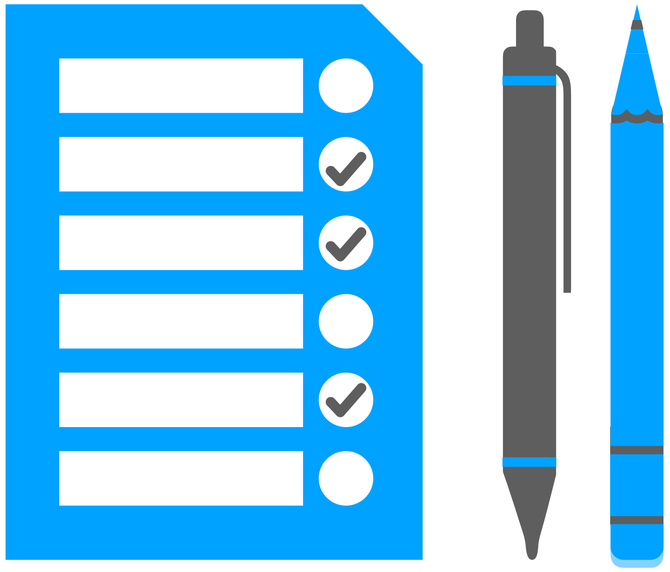
Conditional Statements



- Understand flow of execution
- Understand using Conditional statement to branch out the flow of execution.
- If statement , If else statement
- If else if else statements
- Ternary operator
- Switch statement

Cybertek

After today's session you should be able to:



- Use conditional statement to branch out the code
- Create simple program that take different input and execute different flow according to condition

Cybertek

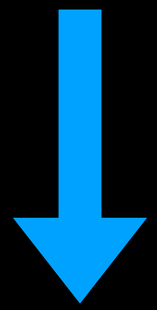
Normal Execution of program

- A program will always execute from top to bottom order in main method unless stated otherwise by programmer

Cybertek

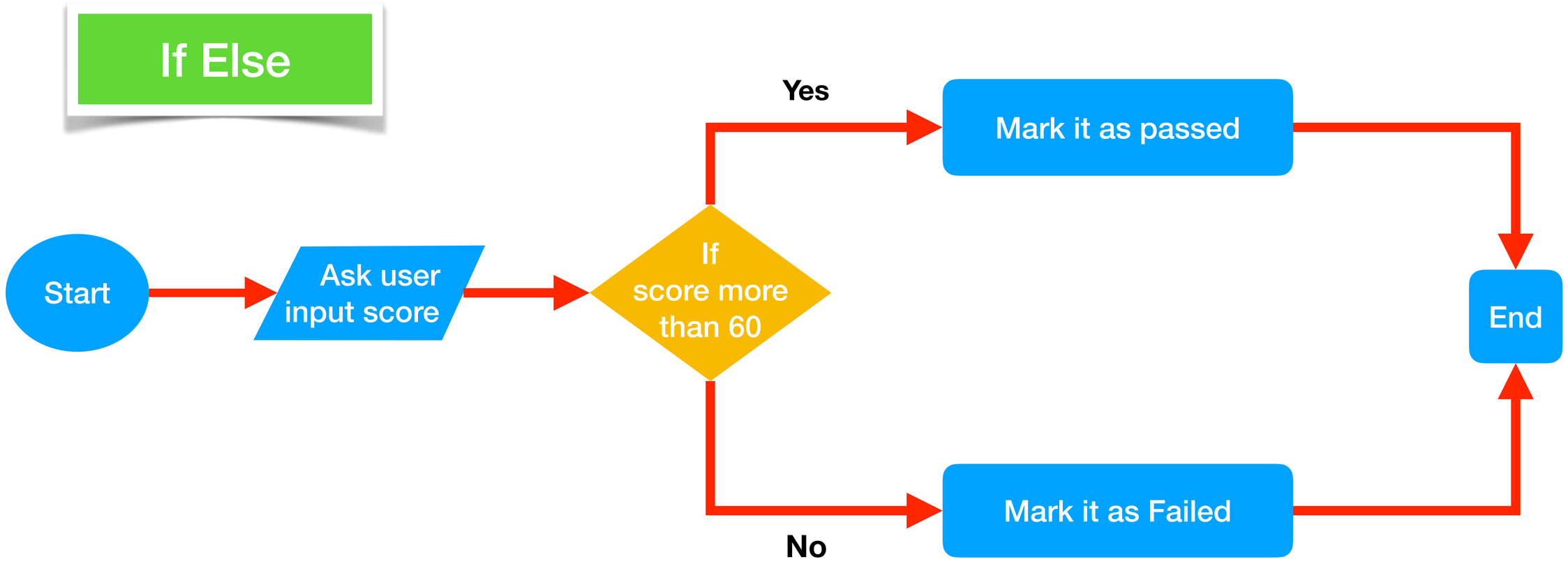
Normal Execution of program

```
class SimpleProgram {

    public static void main(String[] args) {
        // statement 1
        // statement 2
        // statement 3
        // statement 4
    }

}
```



Cybertek

Real life examples of condition



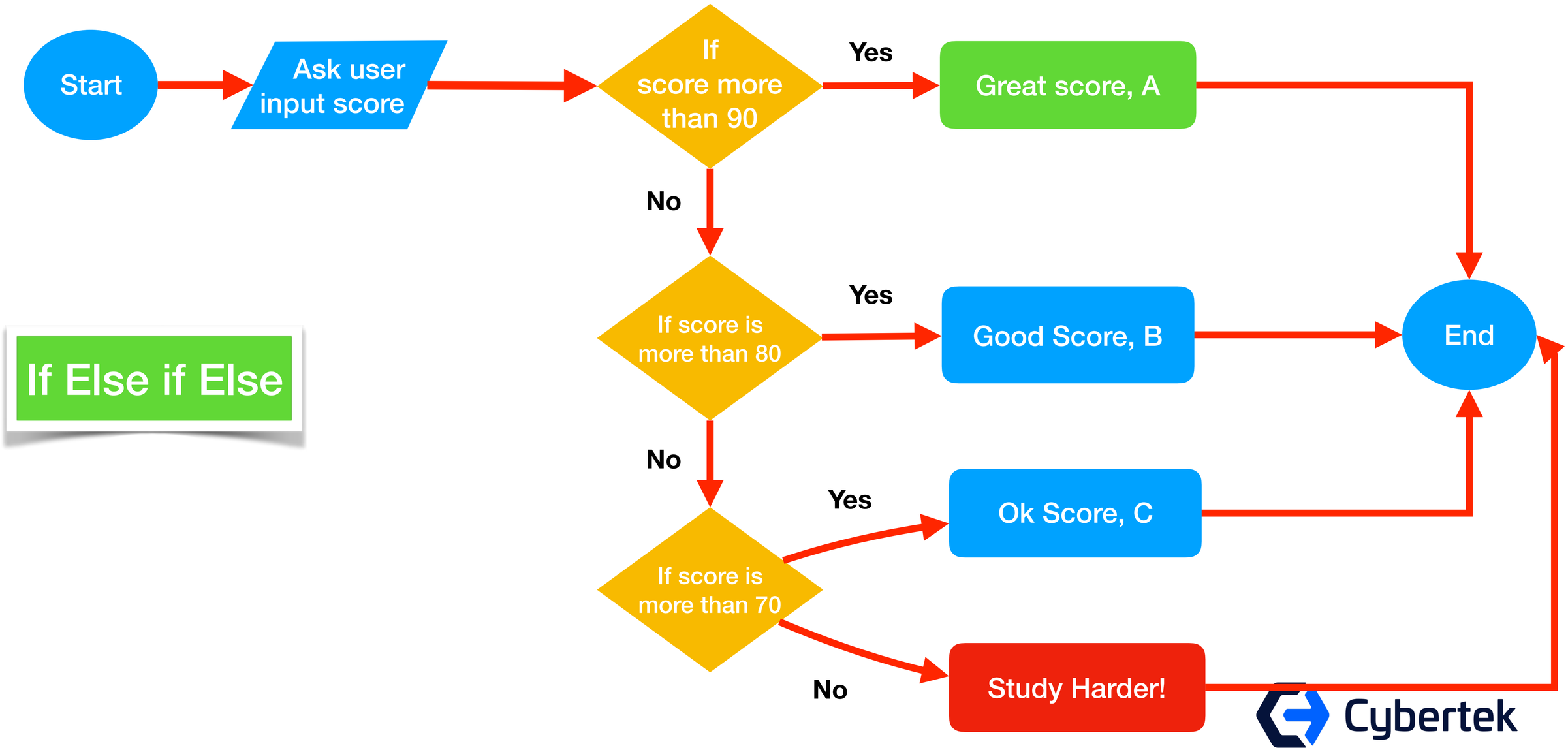
Cybertek

If Else Code example

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

int score = scan.nextInt();
if(score>60) {
  System.out.println(“Passed the exam);
}else {
  System.out.println(“Failed the exam);
}
```

Cybertek

Multi branch if example (if else if else)



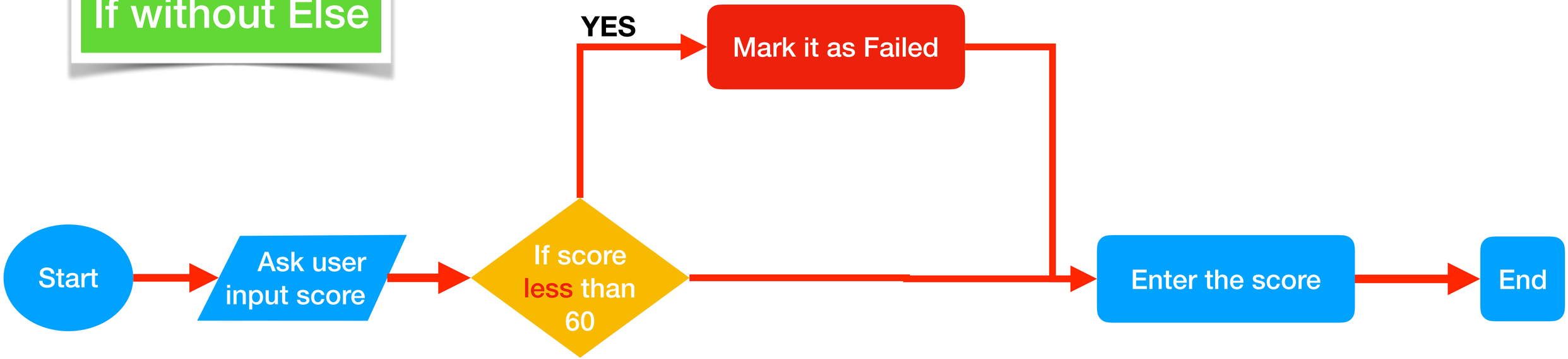
If else if else example

```
if (score > 90) {
  System.out.println("Great score : A!");
 } else if (score > 80) {
  System.out.println(" Good score : B !!");
 } else if (score > 70) {
  System.out.println(" OK SCORE : C !!");
 } else {
  System.out.println("STDUY HARDER!!!!!");
 }
```

Cybertek

Examples of condition

If without Else



No Else block , program just move on if condition is false
Else is not required if no special action needed

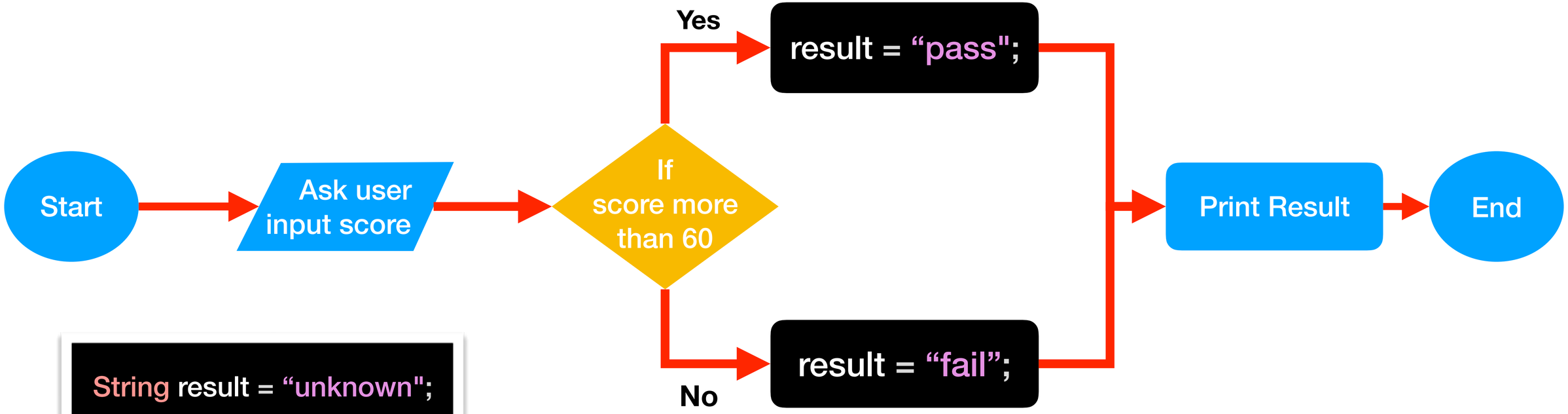
Cybertek

If without else example

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

int score = scan.nextInt();
if(score<60) {
  System.out.println(“Failed the exam);
}

System.out.println(“Entering exam result);
```

Cybertek

Conditional value assignment



Cybertek

Conditional value assignment

```
 Scanner scan = new Scanner(System.in);
String result ;
int score = scan.nextInt();
if(score>60) {
  result = "pass" ;
}else {
  result = "fail";
}
System.out.println("Exam result is "+result);
```

Cybertek

Ternary Operator

```
String result ;
int score = 78

if(score>60) {
  result = "pass";
}else {
  result = "fail";
}
```

```
String result ;
int score = 78

result = (score>60) ? "pass" : "fail";
```

NO

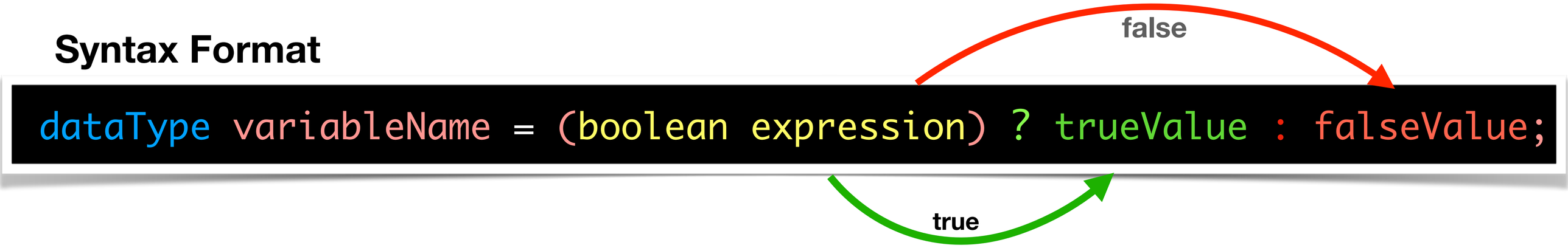
TERNARY OPERATOR

YES

Ternary operator use question mark (?) and colon (:)
Assigned value must be same type as variable type
These two codes are doing exactly same thing

Ternary Operator

Syntax Format



Cybertek

Ternary Operator

Syntax Format

Examples

```
String result = (score>60) ? "pass" : "fail";
```

```
int x = (quality.equals(“good”) ? 100 : 0 ;
```

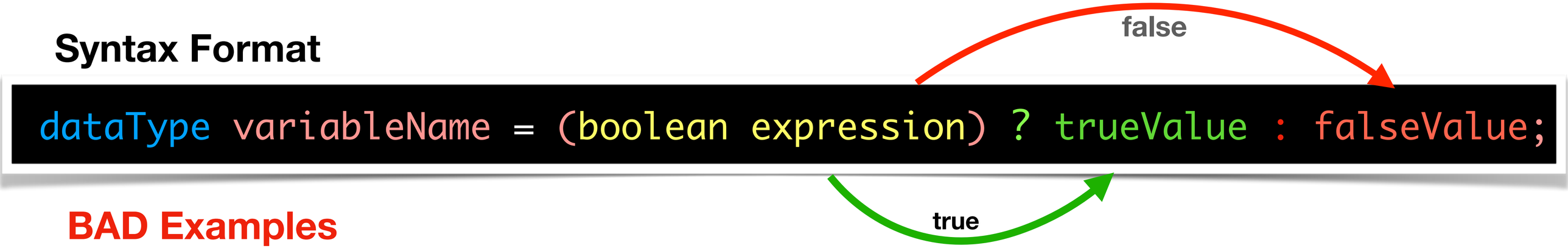
```
char grade = (score>90) ? 'A' : ‘B' ;
```

```
String evenOdd = (score%2==0) ? "even" : "odd";
```

Cybertek

Ternary Operator

Syntax Format



BAD Examples

```
String result = (score>60) ? 'P' : “fail";
```

```
int x = (quality.equals(“good”) ? 100 : 0.5 ;
```

```
char grade = (score>90) ? 'A' : “B” ;
```

```
String evenOdd = (score%2==0) ? 123 : 22.9;
```

Cybertek

Switch statement

Switch statement is used for evaluating **equality** of certain value in multiple case and perform action accordingly

Every switch statement can be done in if else if else statement.

Switch statement make it more readable and easier to maintain

Cybertek

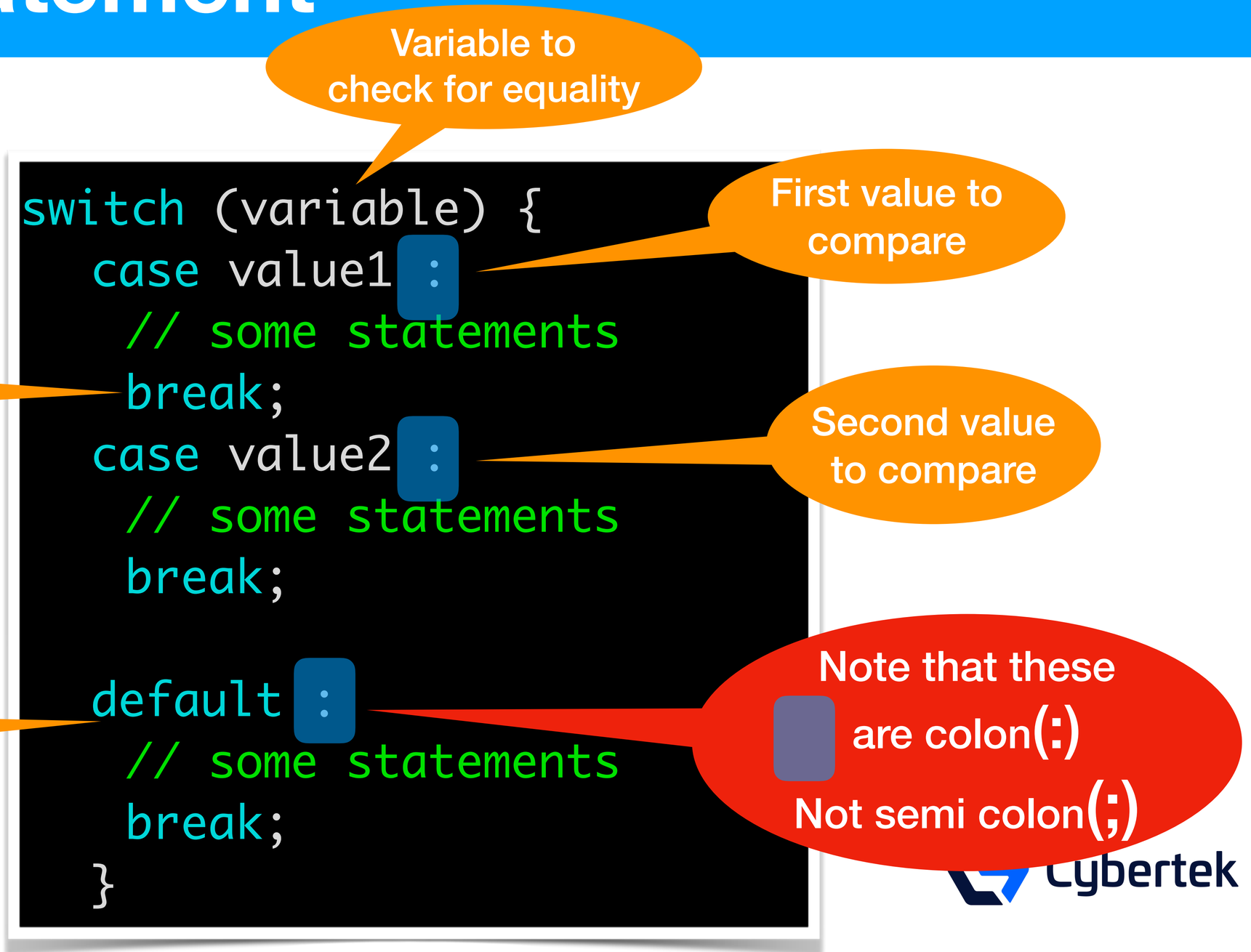
Switch statement

Switch syntax format

```
switch (variable) {
   case value1:
     // some statements
     break;
   case value2:
     // some statements
     break;

   default:
     // some statements
     break;
   }
```

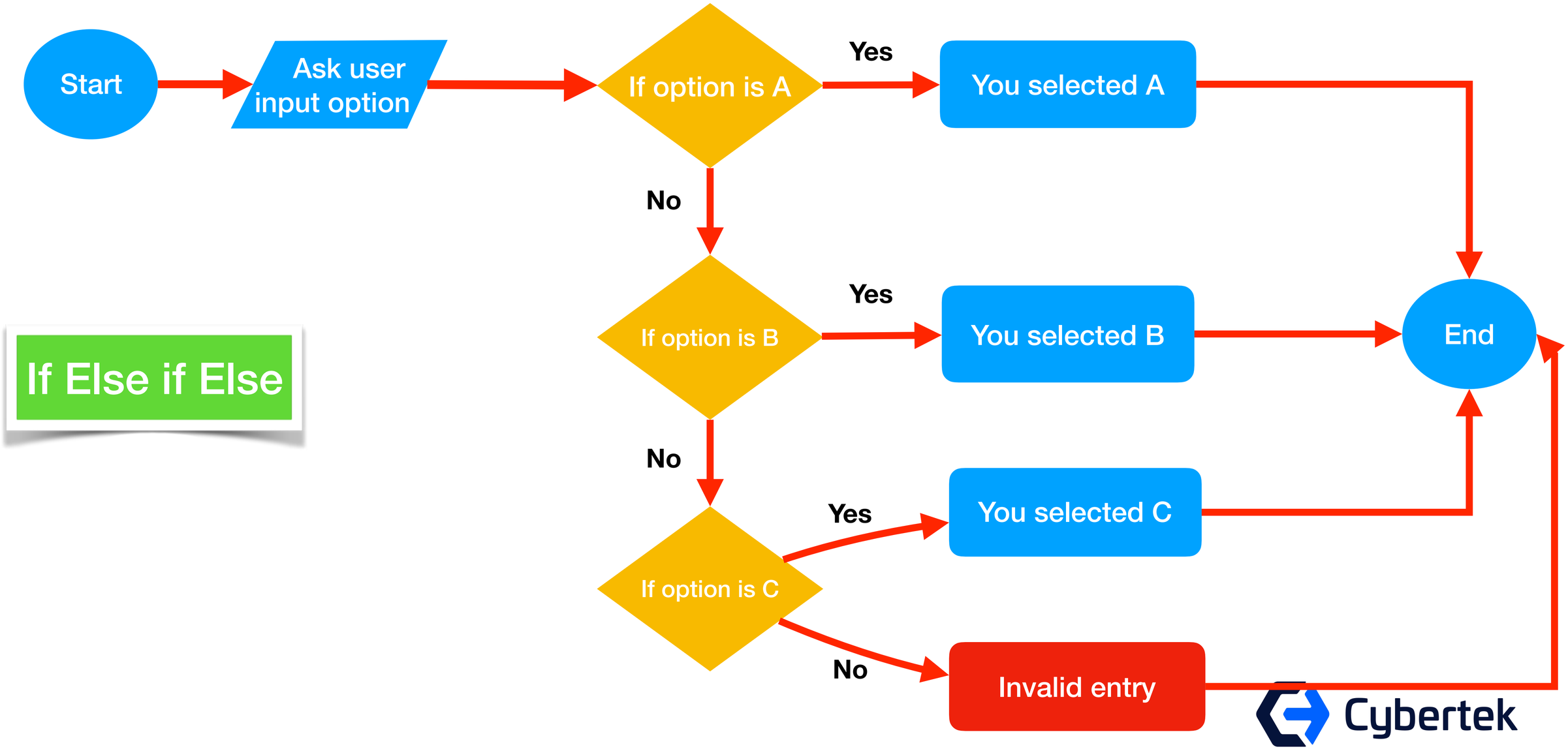
Cybertek

Switch statement



Cybertek

Multi branch if example (if else if else)



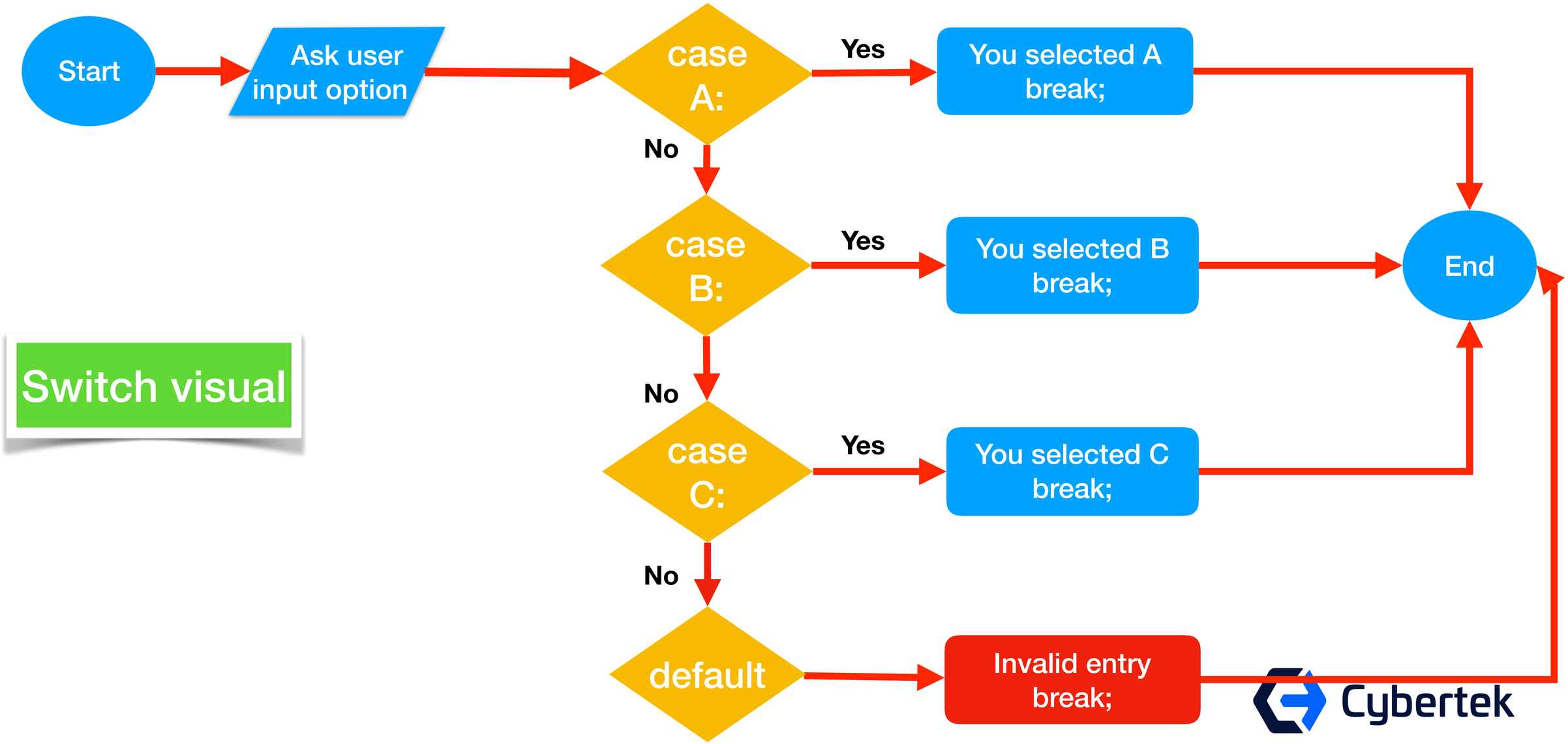
Conditional if else statement

```
char option = 'A';

if (option == 'A') {
  System.out.println("You selected A");
} else if (option == 'B') {
  System.out.println("You selected B");
} else if (option == 'C') {
  System.out.println("You selected C");
}else {
  System.out.println("INVALID ENTRY");
}
```

Simple option selection scenario done using if else if else statement

Cybertek

Switch Statement



Switch visual

Cybertek

Switch statement

Simple option selection scenario done using switch statement

```
 char option = 'A';
switch (option) {
case 'A': //same as if(options=='A')
  System.out.println("You selected A");
  break; //used to exit switch statement
case 'B': //same as if(options=='B')
  System.out.println("You selected B");
  break;
case 'C': //same as if(options=='C')
  System.out.println("You selected C");
  break;
 //same as else part of if statement
 default:
    System.out.println("INVALID ENTRY");
  break;
}
```

Conditional if else statement

```
char option = 'A';

if (option == 'A') {
  System.out.println("A is correct");
} else if (option == 'B'||option == ‘C') {
  System.out.println("Try watching java short 15-18");
} else if (option == 'D') {
  System.out.println(“Incorrect answer);
}else {
  System.out.println("INVALID ENTRY");
}
```

A scenario
to illustrate
A is only correct answer
B and C has same explanation
D is incorrect

Cybertek

Switch statement

Simple option selection scenario done using switch statement

```
   char option = 'A';
switch (option) {
  case 'A': //same as if(options=='A')
    System.out.println("A is correct");
    break; //used to exit switch statement
  case 'B': //same as if(options==‘B’||options==‘C’)
  case 'C':    // take same action
    System.out.println("Try watching java short 15-18);
    break;
  case 'D': //same as if(options=='A')
    System.out.println("Incorrect answer");
    break; //used to exit switch statement
  default:
      System.out.println("INVALID ENTRY");
    break;
 }
```

Switch statement without break

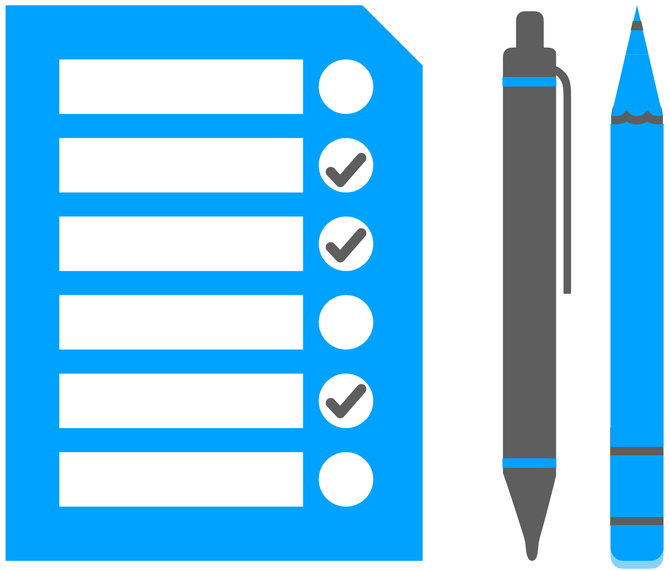
break is required to break out of switch when match found

If break missing , it will just execute the rest of the cases.
Its called fall-through.

Output of the program :
You selected B
You selected C
INVALID ENTRY

```
 char option = ‘B’;
switch (option) {
  case 'A': //same as if(options=='A')
    System.out.println("You selected A");
  case 'B': //same as if(options=='B')
    System.out.println("You selected B");
  case 'C': //same as if(options=='C')
    System.out.println("You selected C");
    //same as else part of if statement
   default:
      System.out.println("INVALID ENTRY");
  }
```

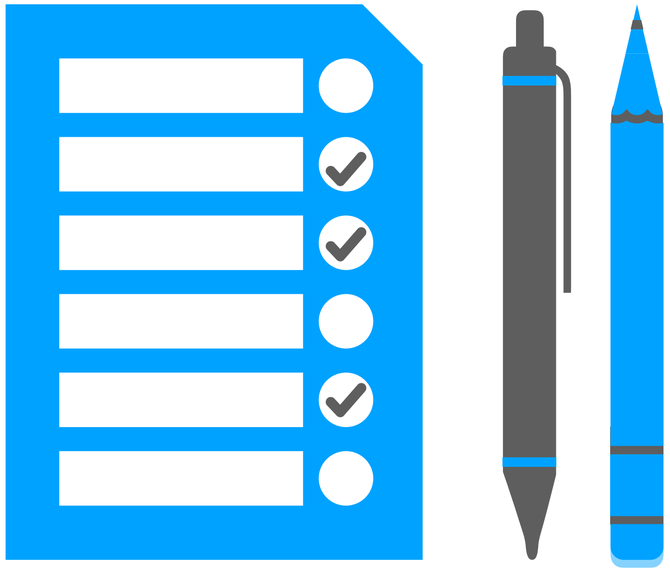
String Class



- Understand the concept of class and object
- Understand ways to create String objects from String class
- Understand the actions we can take using String objects
- Practice and use available String methods

Cybertek

After today's session you should be able to:

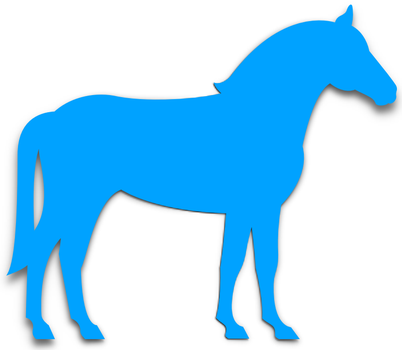


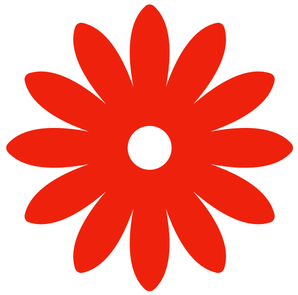
- Use conditional statement to branch out the code
- Create simple program that take different input and execute different flow according to condition

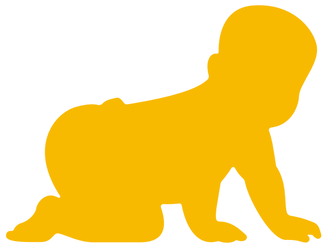
Cybertek

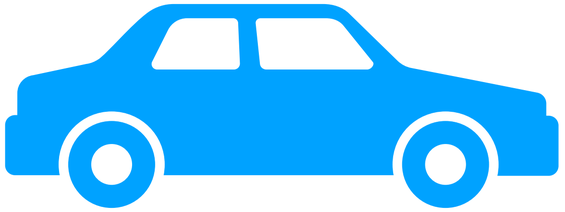
What is an Object

What is an Object

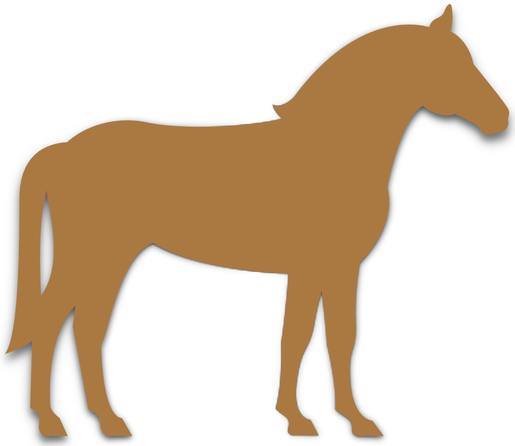








Object has attributes



Breed : mustang

Height : 1.5m

Color : dark brown

Cybertek

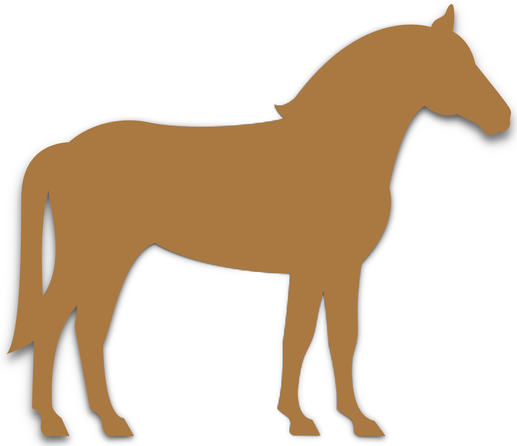
Object has attributes

Year : 2018

Make : Honda

Model : Accord

Object has behavior



Run

Eat

Sleep

Object has behaviors

drive

changeDirection

MakeNoise

Object behaviors

Some actions can be performed directly without extra instruction

Some other action need external data / extra instruction while performing the actions

Cybertek

Object behaviors

Below is real life example and how it might look like in Java:

```
Tell me the length of this table  —>  tableObject.length() ;
Display the mileage on the car —>  carObject.displayMileage() ;

Change direction of the car to the right —-> carObject.changeDirection( “right”) ;
Go run 5 miles  —> personObject.runMiles( 5 ) ;
Study subject Java.  — —>  studentObject.study(“java”)
Study with full dedication.  — —>  studentObject.studyWithFullDedication(true)

Tell me your first name and last name — —> personObject.showName(“john”, “adam”)
Study java for 5 hours today at library — —> studentObject.study(“java”, 5 , “Library”)
```

Cybertek

Object behaviors

Some actions generate a result after execution
This result can be saved and used for later or directly used after performing actions

Some actions does not generate any result , it purely perform an action

Object behaviors

Below is real life example and how it might look like in Java:

Tell me the length of this table —> tableObject.length() ; —> return number
Display the mileage on the car —> carObject.displayMileage() ; —> just perform display action without returning a result

Change direction of the car to the right —-> carObject.changeDirection(“right”) ;
—> Just action without returning/generating a value
Tell me one character in your name —> personObject.getChar(1) ; —> return character

Cybertek

Method

In java, these behaviors / actions are called method

Performing the action on an object is called executing a method OR calling a method

Question :

- Where are these objects come from ?
- Where are these methods defined ?
- How do we know it takes or does not take any external date/aruguments ? If it does , how do we know what kind of data ?
- How do we know it returns a value or not ? If it does , how do we know the return type?

Answer :

Class

What is a class

What is an class

A blueprint/template to create an object

Each objects are created out of the corresponding class

What is an class

Each objects are created out of the corresponding class

That's why anything that not primitive, we call it object type or class type or reference type. They are all referring same thing

Cybertek

What is an class

Whenever a class is created , it can be a data type
For example
String has it own class —> String s ;

If you have *public class Car {} —-> Car c ;*

Cybertek

Inside class

A class define properties and behaviors of object

Every object created out of this template/blueprint will have all the properties and behaviors

String class

String Class

A special class in java to create and manipulate strings.

```
String s = "Hello World" ;
```

String Object

Object that represent A sequence of characters

"Hello World" ;

2 way to create String object

- **String Literal : created using quotation directly**

```
String s = "Hello World" ;
```

- **Using new keyword**

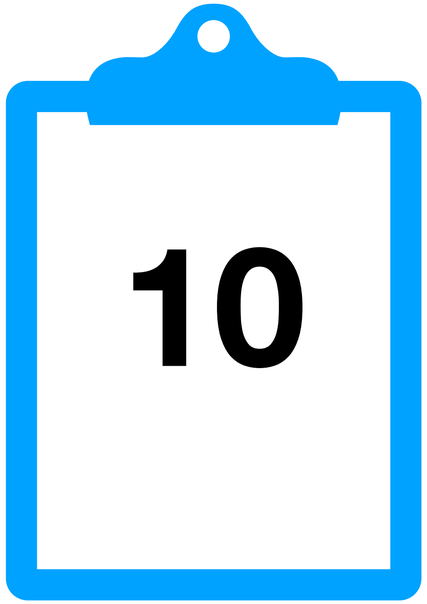
```
String s = new String(“Hello World”) ;
```

Cybertek

How Primitives and Objects are stored in Memory

Primitive Types

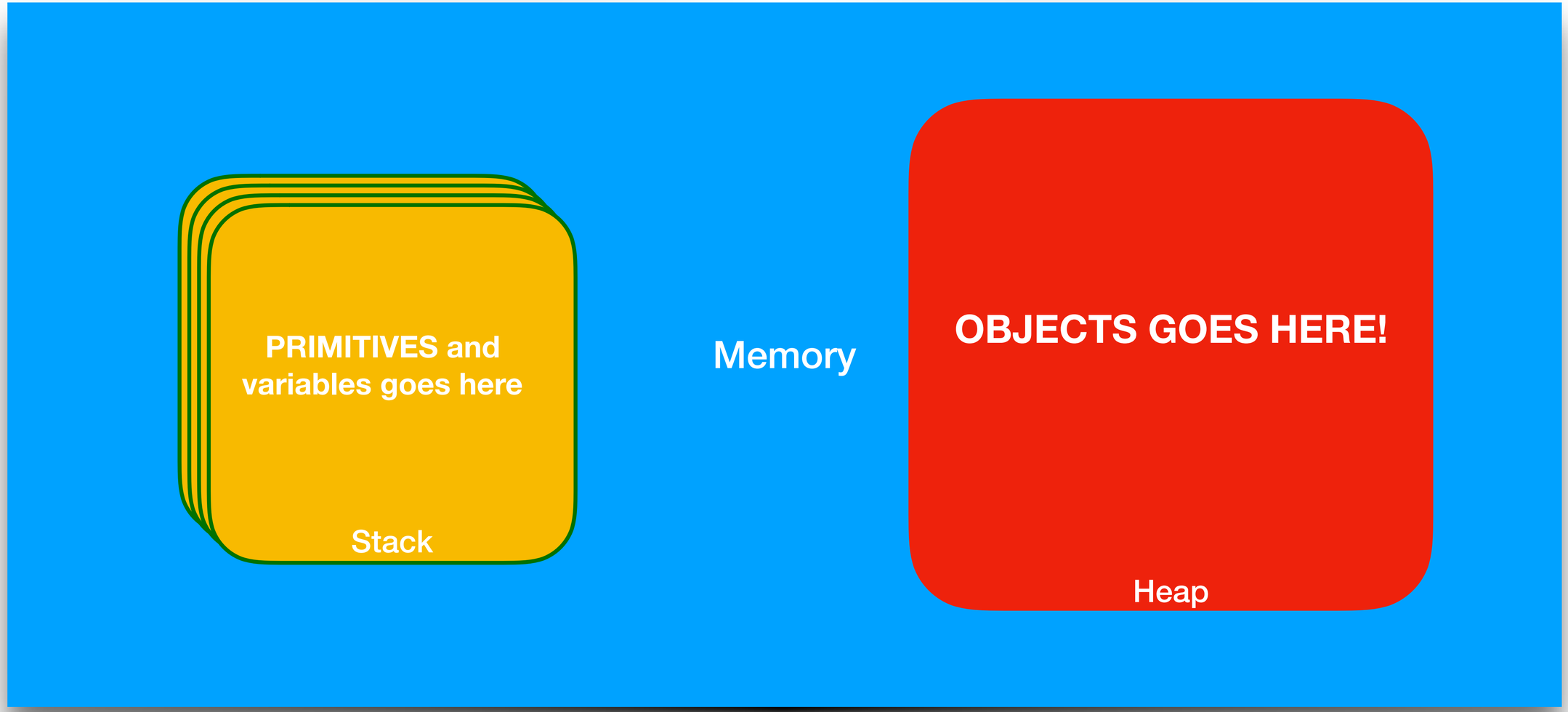
```
int a = 10 ;
```



a is a container/ piece of memory that can store int number directly inside
All primitive values are directly stored in the container , that's why it's also called value type

Cybertek

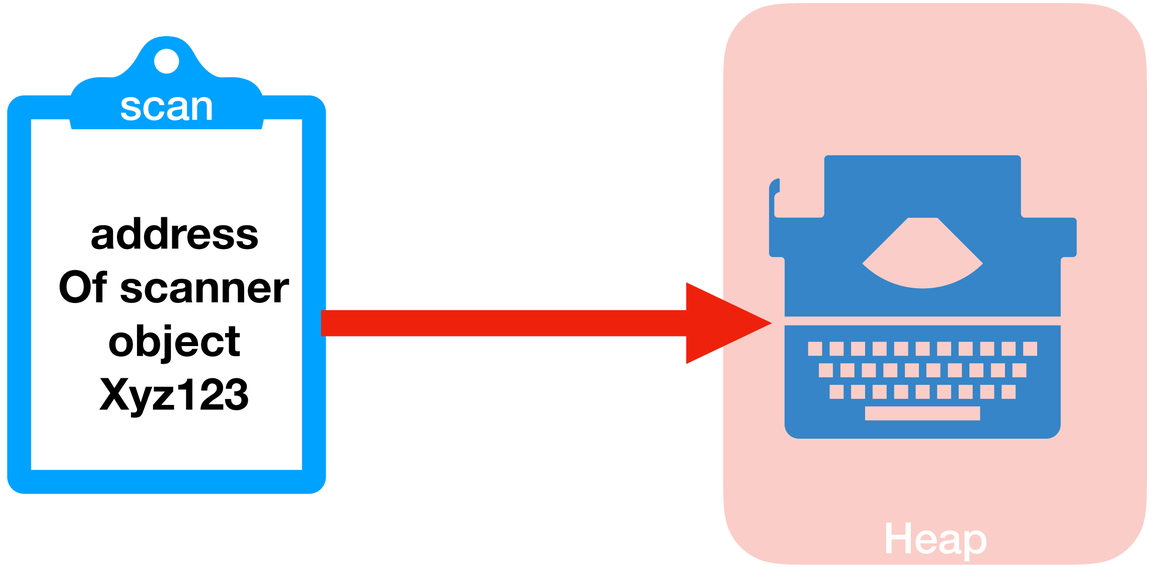
Stack and Heap



Cybertek

Object Types/Class Type/Reference Type

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

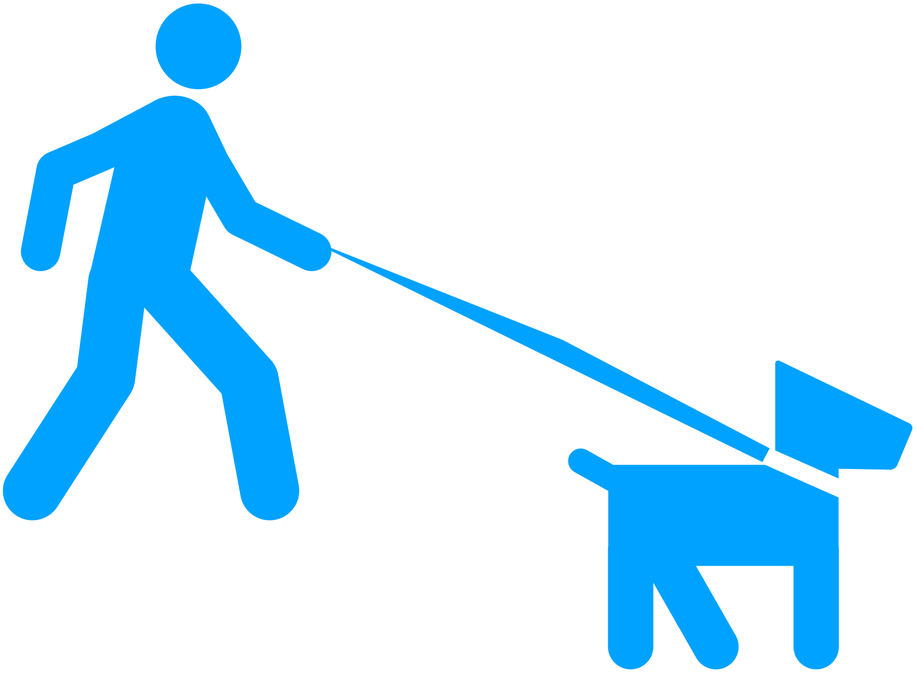


scan is a container/ piece of memory that can store address of any single scanner

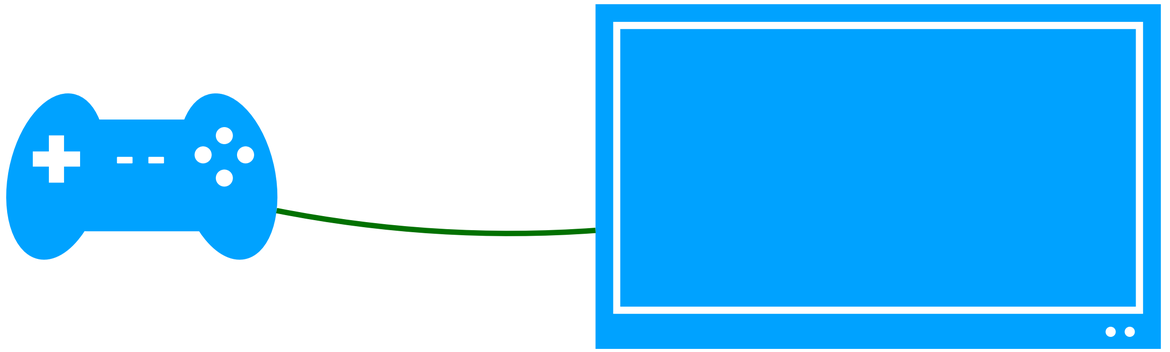
All non-primitive type variables store only address of the object in memory
Or a reference/pointer to actual object in memory.
That's why it's also called Reference Type

Cybertek

More example of reference type



A dog leash has a pointer to the dog
And it can control the dog

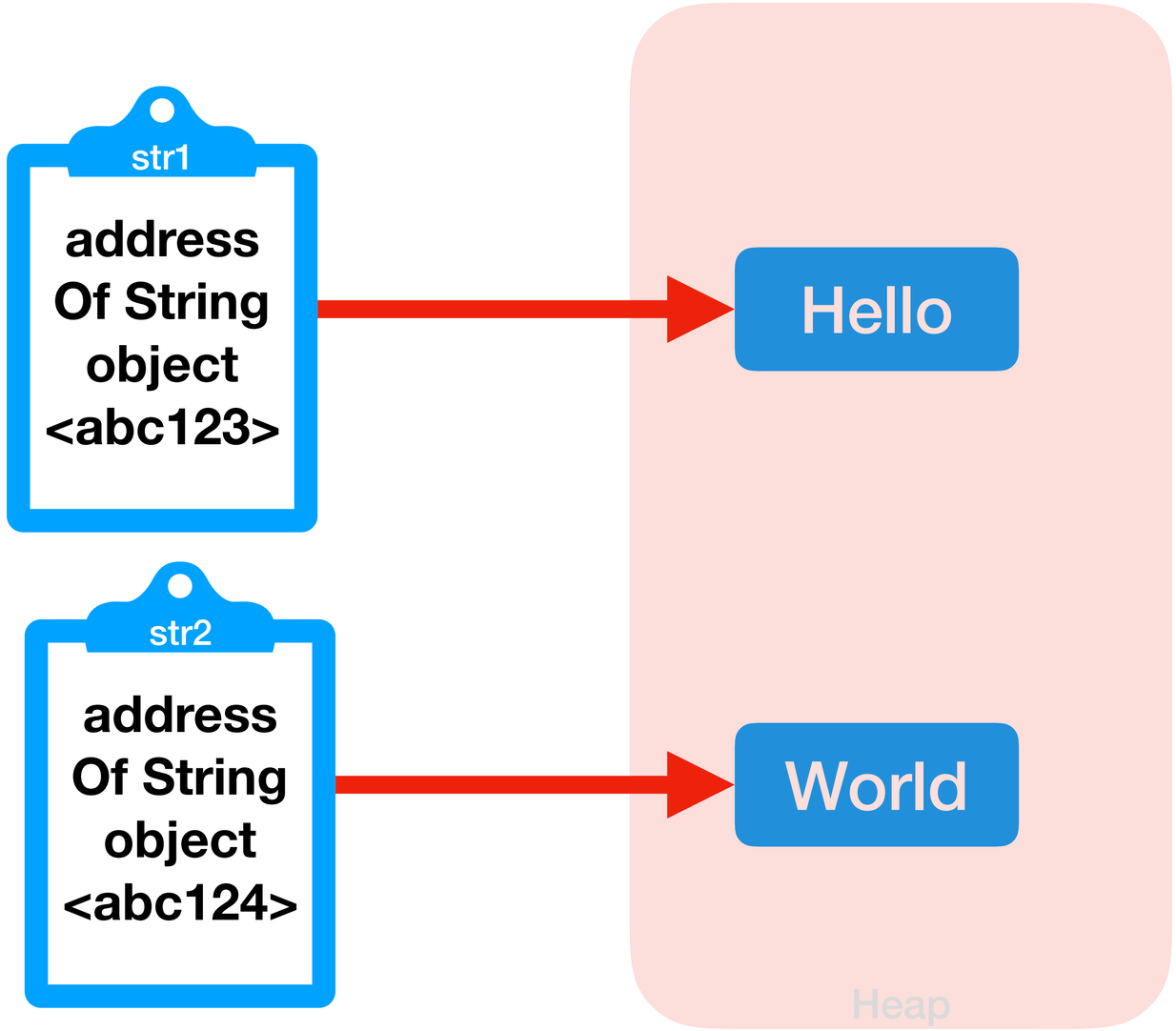


A remote has reference/pointer to the TV
And it can be used to control the tv remotely

Cybertek

String Example

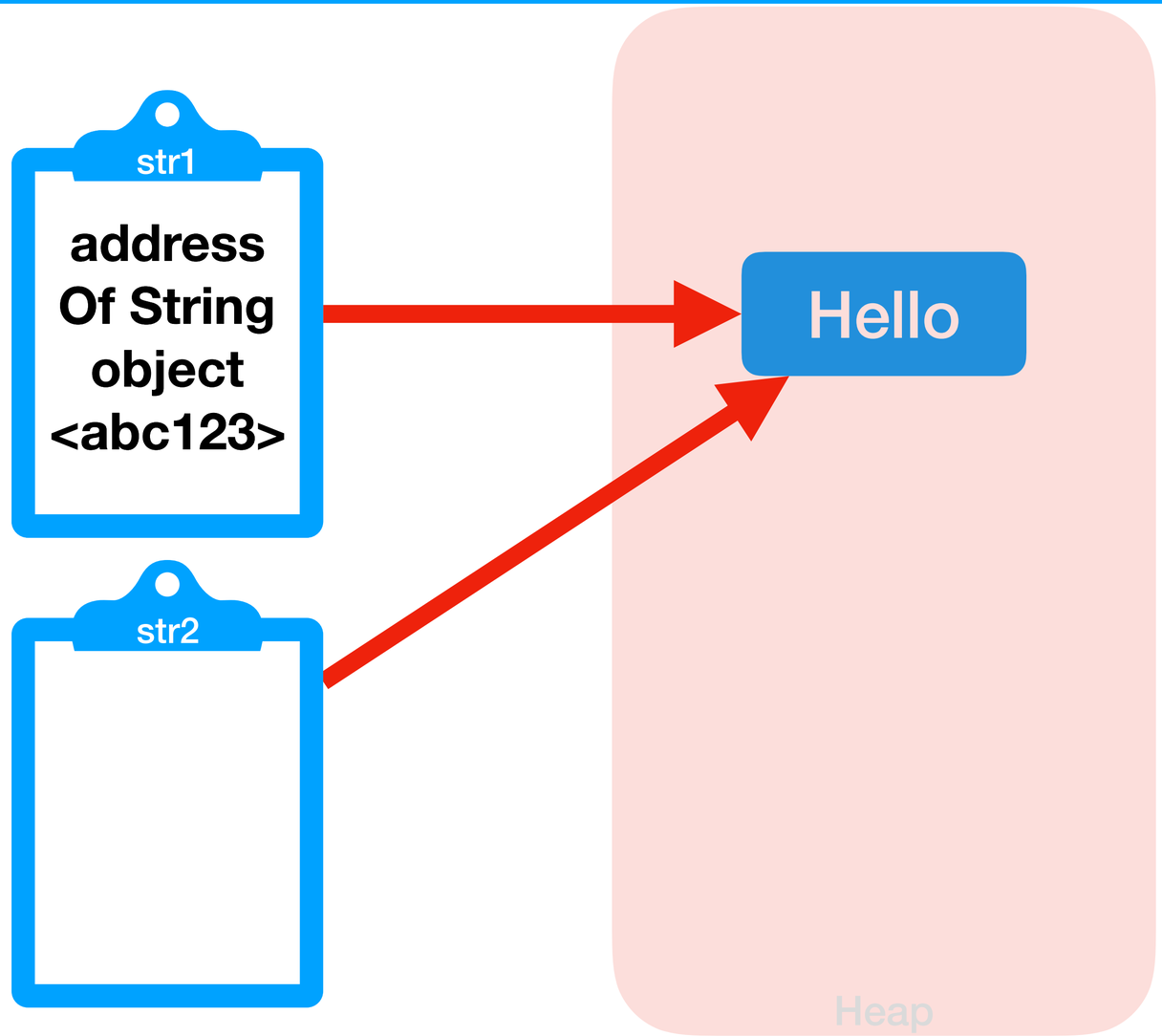
```
String str1 = “Hello";
String str2 = “World";
```



Cybertek

String Example

```
String str1 = “Hello";
String str2 = str1;
print(str1==str2) —> true
```

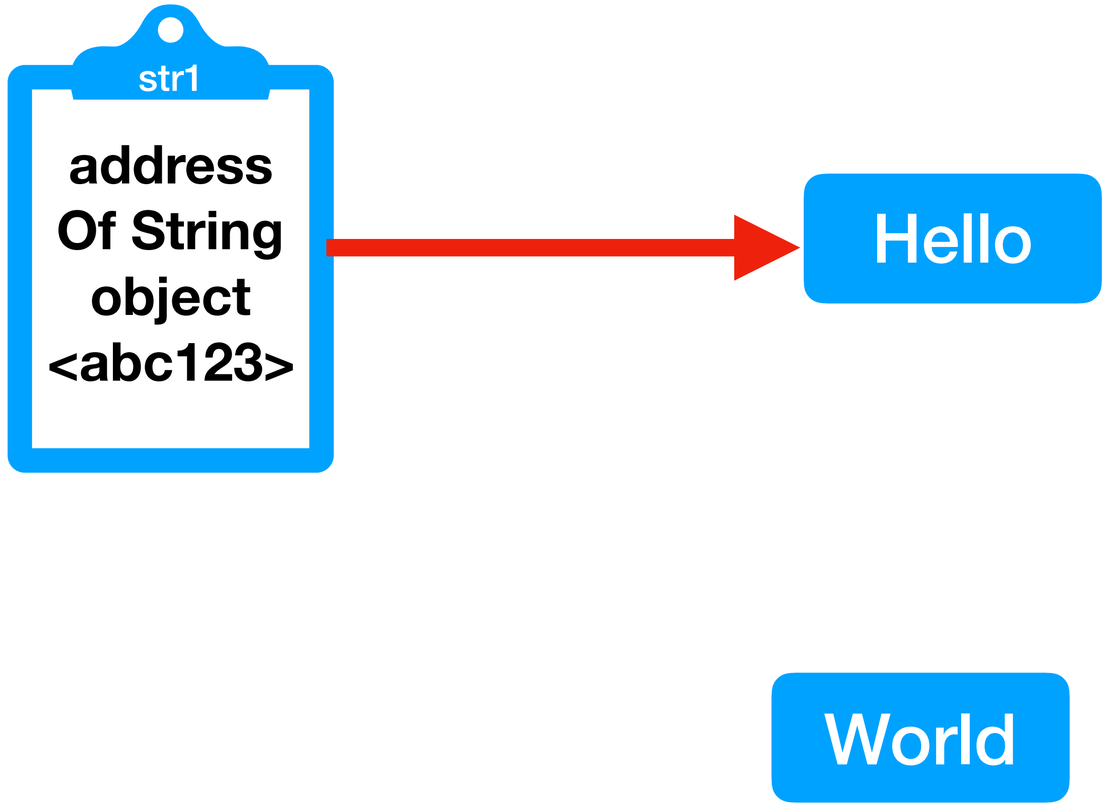


Cybertek

String is immutable
Once created can not be changed

String Example

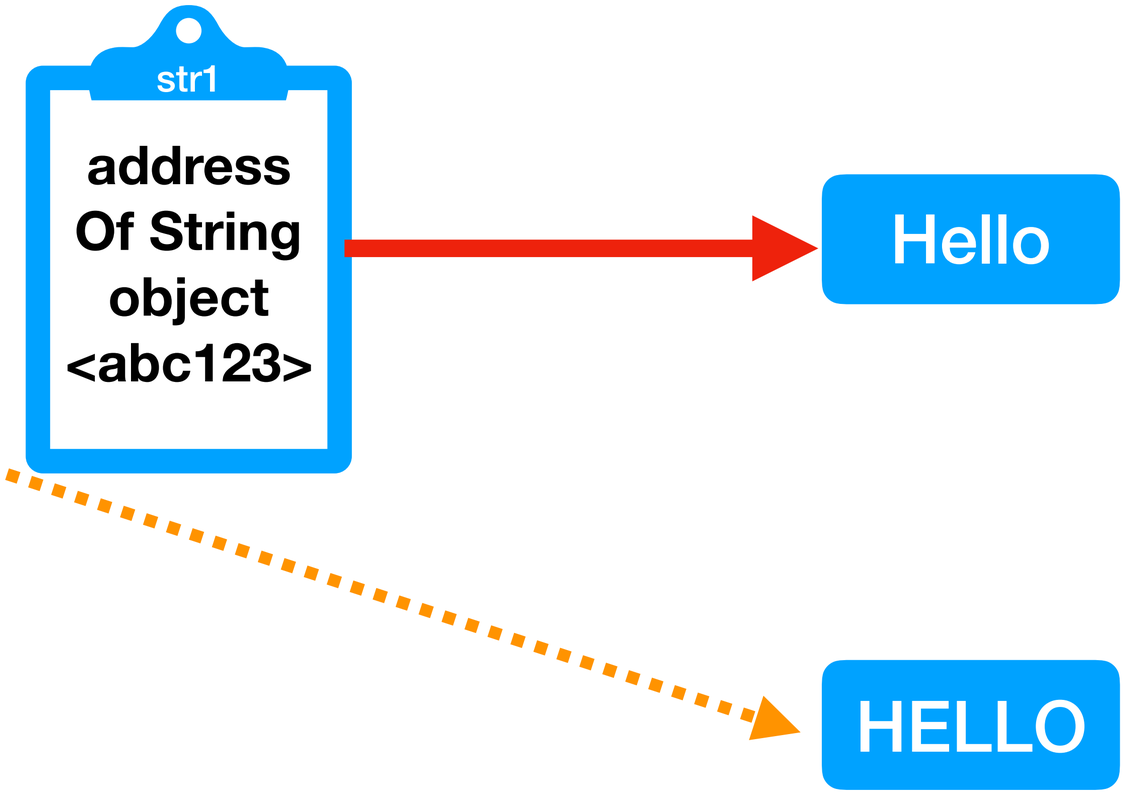
```
String str1 = “Hello";
      str1 = “World";
```



Cybertek

String Immutability example

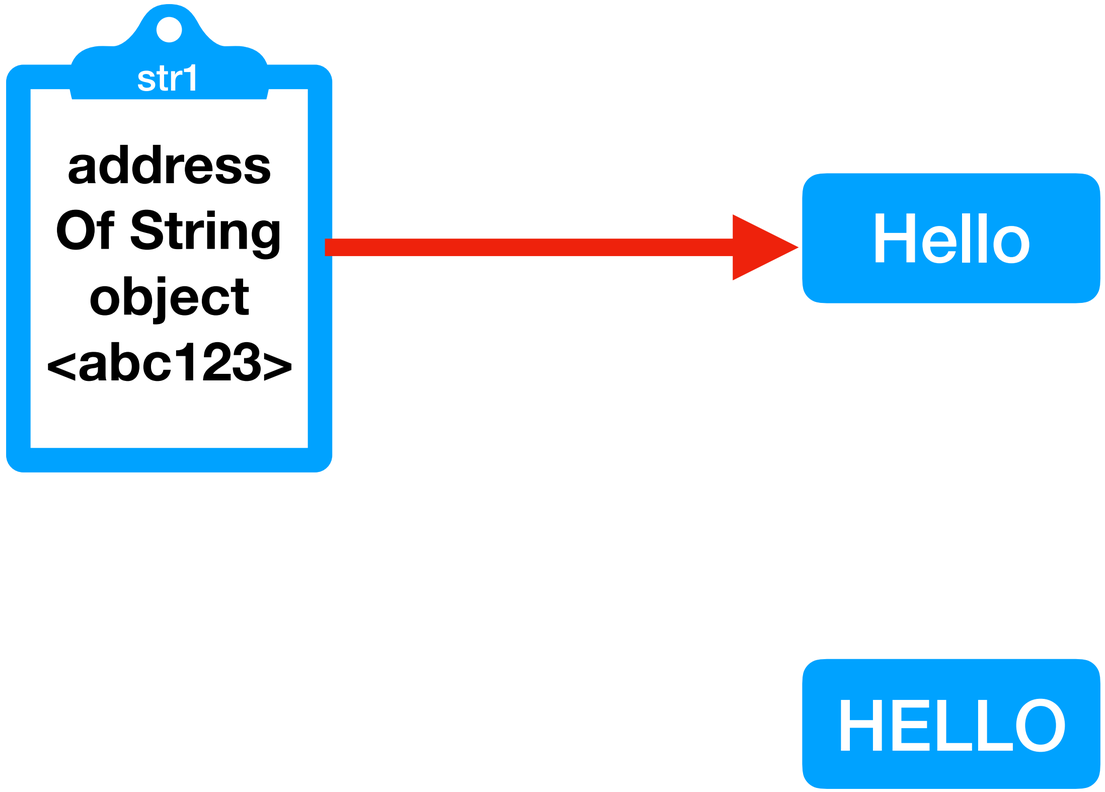
```
String str1 = “Hello";
          str1.toUpperCase();
print(str1)  //—> Hello
```



Cybertek

String Immutability example

```
// Correct Way
String str1 = “Hello";
    str1 =   str1.toUpperCase();
print(str1)  //—> “HELL0”
```

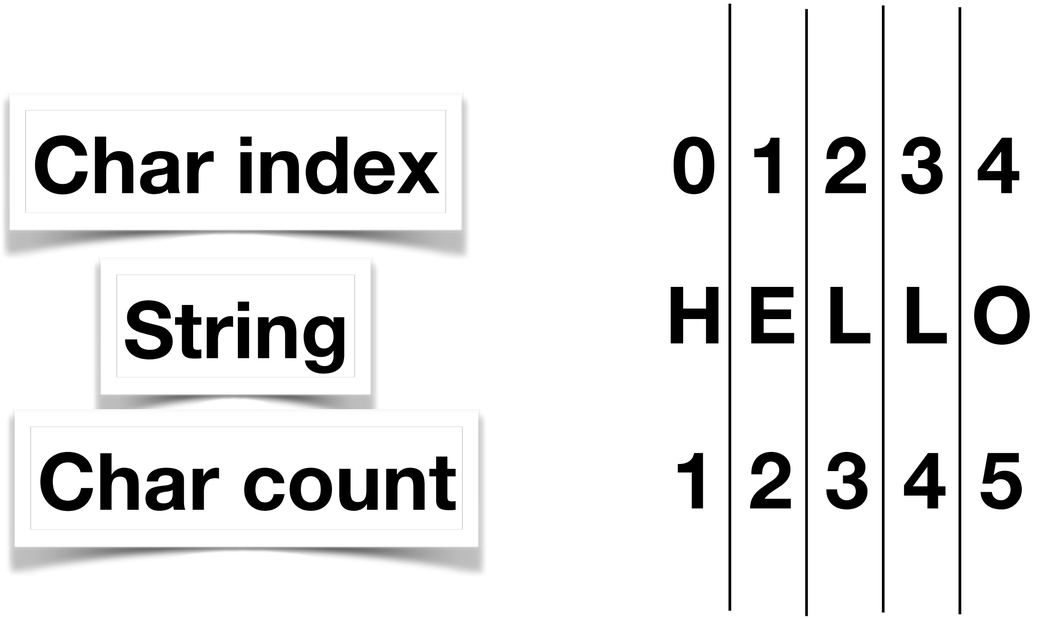


Cybertek

String methods /behaviors

Actions that we can take using string object

String anatomy




```
String str = “Hello”;
```

Cybertek

String methods : length()

Char index	0	1	2	3	4
String	H	e	l	l	0
Char count	1	2	3	4	5

Last index is 4

Character count is 5

```
String str = “Hello”;
int charCount = str.length();
```

length() method return the count of character

Cybertek

charAt(index)

```
 returns char value for the
particular index

 char c1 = str.charAt(0); //-> H
 char c5 = str.charAt(4); //-> o
 char co = str.charAt(20); //
      -> Exception at runtime
```

Char index	0	1	2	3	4
String	H	e	l	l	0
Char count	1	2	3	4	5

Cybertek

equals(anotherStr)

```
 returns true if two string are
equals , false if not

 boolean b1 = str.equals("Hello");//->true
 boolean b1 = str.equals(“abc");//->false
```

Char index	0	1	2	3	4
String	H	e	l	l	0
Char count	1	2	3	4	5

Cybertek

equalsIgnoreCase(anotherStr)

```
 returns true if two string are
equals , false if not

 boolean b1 =
str.equalsIgnoreCase("hello");//->true
 boolean b2 =
str.equalsIgnoreCase(“HELLO”);//->true
```

Char index	0	1	2	3	4
String	H	e	l	l	0
Char count	1	2	3	4	5

Cybertek

toLowerCase()

```
 returns lowercase String (without
changing original String)

String str2 = str.toLowerCase();
//->hello
```

Char index	0	1	2	3	4
String	H	e	l	l	0
Char count	1	2	3	4	5

Cybertek

toUpperCase()

```
 returns uppercase String (without
changing original String)

String str2 = str.toUpperCase();
//->HELLO
```

Char index	0	1	2	3	4
String	H	e	l	l	0
Char count	1	2	3	4	5

Cybertek

contains(anotherStr)

```
 returns true or false after
matching the sequence of char value.

boolean b3 =str2.contains("He");  //->true
boolean b3 =str2.contains(“abc"); //->false
```

Char index	0	1	2	3	4
String	H	e	l	l	0
Char count	1	2	3	4	5

Cybertek

indexOf(anotherStr)

```
 returns the specified substring
index. -1 if not found

int index1 = str.indexOf("llo");  //->2
int index2 = str.indexOf("az");   //->-1
```

Char index	0	1	2	3	4
String	H	e	l	l	0
Char count	1	2	3	4	5

Cybertek

indexOf(anotherStr, fromIndex)

```
 returns the specified substring
index starting with given index.
-1 if not found

int x = str.indexOf("l");  //-> 2
int y = str.indexOf(“l",3);//-> 3 it will
look for first l starting from index 3
int z = str.indexOf(“l”,4) //-> -1
```

Char index	0	1	2	3	4
String	H	e	l	l	0
Char count	1	2	3	4	5

Cybertek

lastIndexOf(anotherStr)

```
 returns the specified substring
index looking from end to beginning
order . -1 if not found

int index1 = str.lastIndexOf("l");  //->3
int index2 = str.lastIndexOf("az");//->-1
```

Char index	0	1	2	3	4
String	H	e	l	l	0
Char count	1	2	3	4	5

lastIndexOf(anotherStr, fromIndex)

```
 returns the specified substring
index looking from end to beginning
order . -1 if not found

int x = str.lastIndexOf("l");  //-> 2
int y = str.indexOf(“l",3);//-> 3 it will
look for first l starting from index 3
int z = str.indexOf(“l”,4) //-> -1
```

Char index	0	1	2	3	4
String	H	e	l	l	0
Char count	1	2	3	4	5

Cybertek

subString(beginningIndex, EndingIndex)

```
 returns substring from given begin
index till right before end index.

String p1 = str.substring(1,3);  //-> el
String p2 = str.substring(0,1);  //-> H
String p3 = str.substring(2,5);  //-> llo
String p4 = str.substring(2,7);
                    -> Exception at runtime
```

Char index	0	1	2	3	4
String	H	e	l	l	0
Char count	1	2	3	4	5

Cybertek

subString(beginningIndex)

```
 returns substring from given begin
index till the end of String.

String p1 = str.substring(1);  //-> ello
String p2 = str.substring(3);  //->lo
String p3 = str.substring(5);
                    -> Exception at runtime
```

Char index	0	1	2	3	4
String	H	e	l	l	0
Char count	1	2	3	4	5

Cybertek

isEmpty()

```
 returns true if string is Empty , false if not

 String str = “";
 boolean b4 = str.isEmpty();//->true
  String str = “Hello";
 boolean b5 = str.isEmpty();//->false
```

Char index	0	1	2	3	4
String	H	e	l	l	0
Char count	1	2	3	4	5

Cybertek

startsWith(anotherStr)

```
 returns true if string start with
give string , false if not

 boolean b1 =str.startsWith(“He");//->true
 boolean b2 =str.startsWith("k");//->false
```

Char index	0	1	2	3	4
String	H	e	l	l	0
Char count	1	2	3	4	5

Cybertek

endsWith(anotherStr)

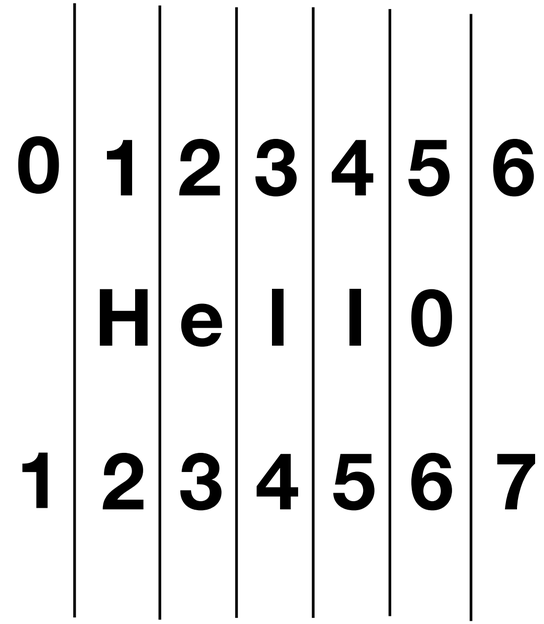
```
 returns true if string start with
give string , false if not

 boolean b1 =str.endsWith("lo");//->true
 boolean b2 =str.endsWith("k");//->false
```

Char index	0	1	2	3	4
String	H	e	l	l	0
Char count	1	2	3	4	5

Cybertek

trim()

```
 removes beginning and ending
spaces of this string.

String str = “ Hello ";
String trimmed = str.trim(); //->true
```



Cybertek

concat(AnotherString)

```
concatenates the specified string.

String str = “Hello”;
String s4 = str.concat(“ World”);
//Hello World
```

Char index	0	1	2	3	4
String	H	e	l	l	0
Char count	1	2	3	4	5

Cybertek

replace(oldChar, newChar)

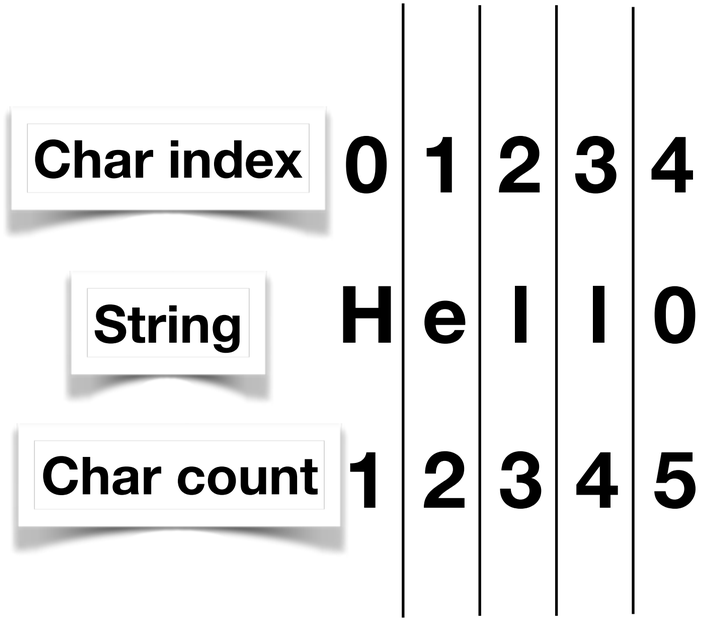
```
 replaces all occurrences of the
specified char value.

 String s4 = str.replace('e','a'); //Hallo
 String s4 = str.replace('l','k'); //Hekko
String s5 = str.replace('z','a'); //Hello
// if not found , it will be just ignored
```

Char index	0	1	2	3	4
String	H	e	l	l	0
Char count	1	2	3	4	5

Cybertek

replace(oldStr, newStr)

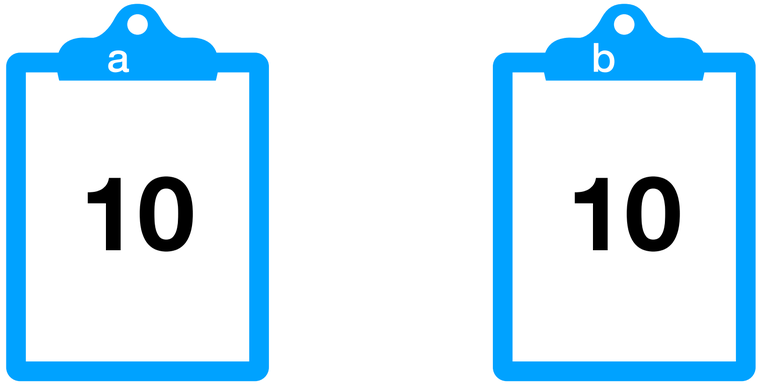
```
 replaces all occurrences of the
specified string value.

 String s4 = str.replace(“ello”,”zz”); //Hzz

String s5 = str.replace(“l”,’YY’); //HeYYYYo
// if not found , it will be just ignored
```



Cybertek

Equality recap with == for PRIMITIVE TYPES

```
int a = 10 ;
int b = 10 ;
```



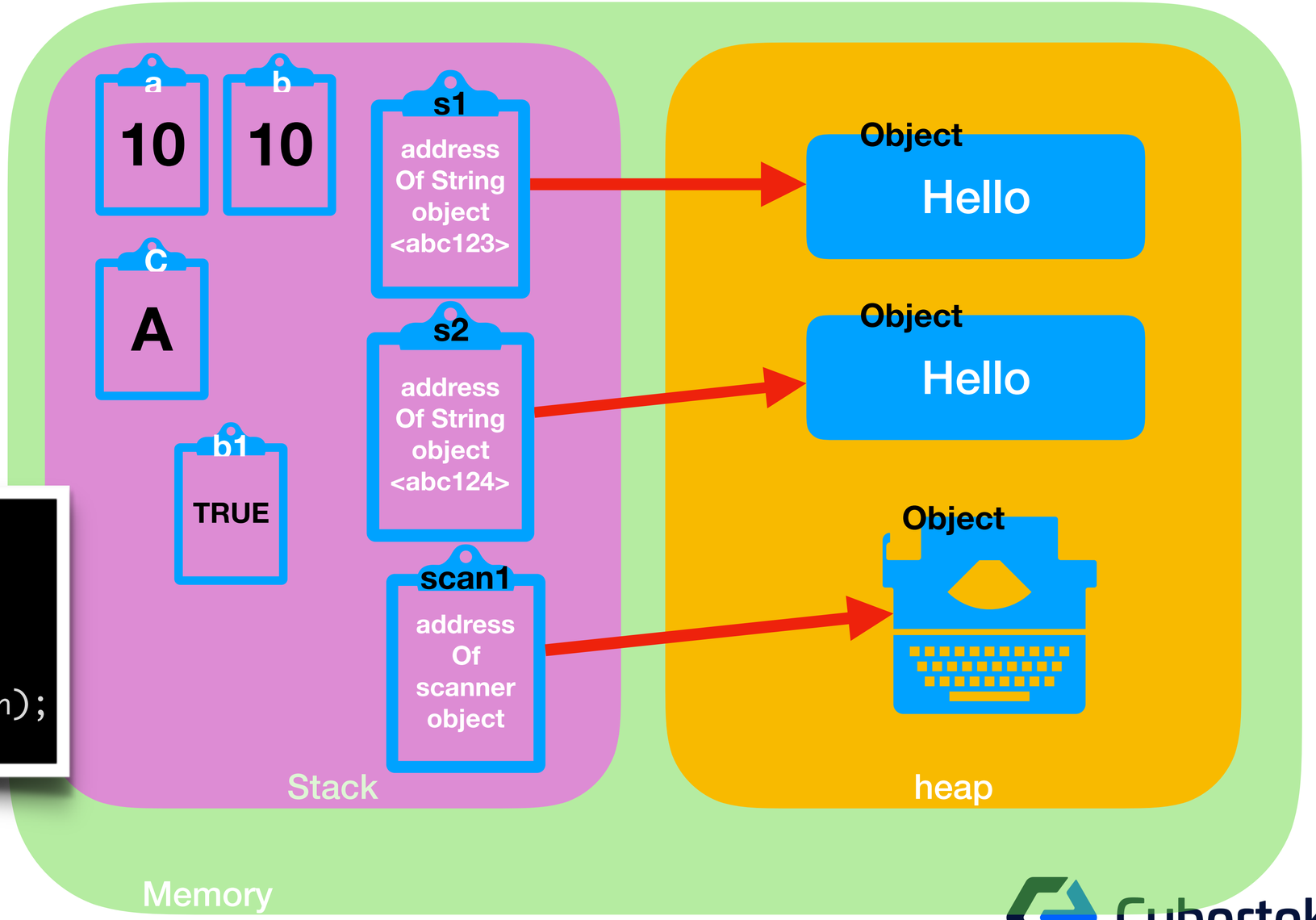
- values are directly stored inside container created while declaring the variables.
- == will always compare pure value stored inside container.
- == should always be used for content check

Cybertek

Stack and heap

```
int a = 10 ;  int b = 10 ;
print(a==b); //—> true
  char c = 'A';
  boolean b1 = true;
```

```
String s1 = new String("Hello") ;
String s2 = new String("Hello") ;
print(s1 == s2 ) ; //—> false
Scanner scan1 = new Scanner(system.in);
```

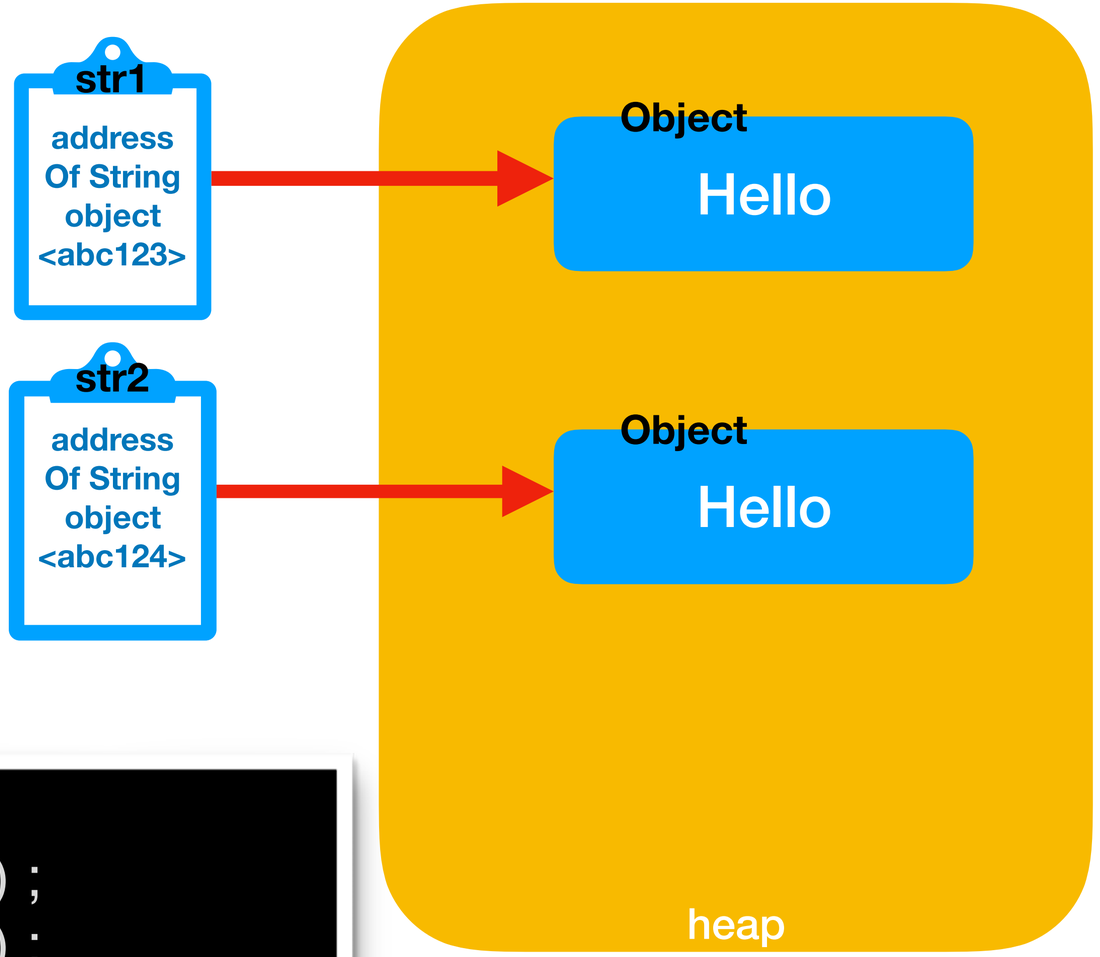


Cybertek

Equality recap with == for Reference type

- objects are not directly stored inside container created while declaring the variables.

- The container only hold the heap address of the object /a reference to that object. If the address is not the same, == always return false



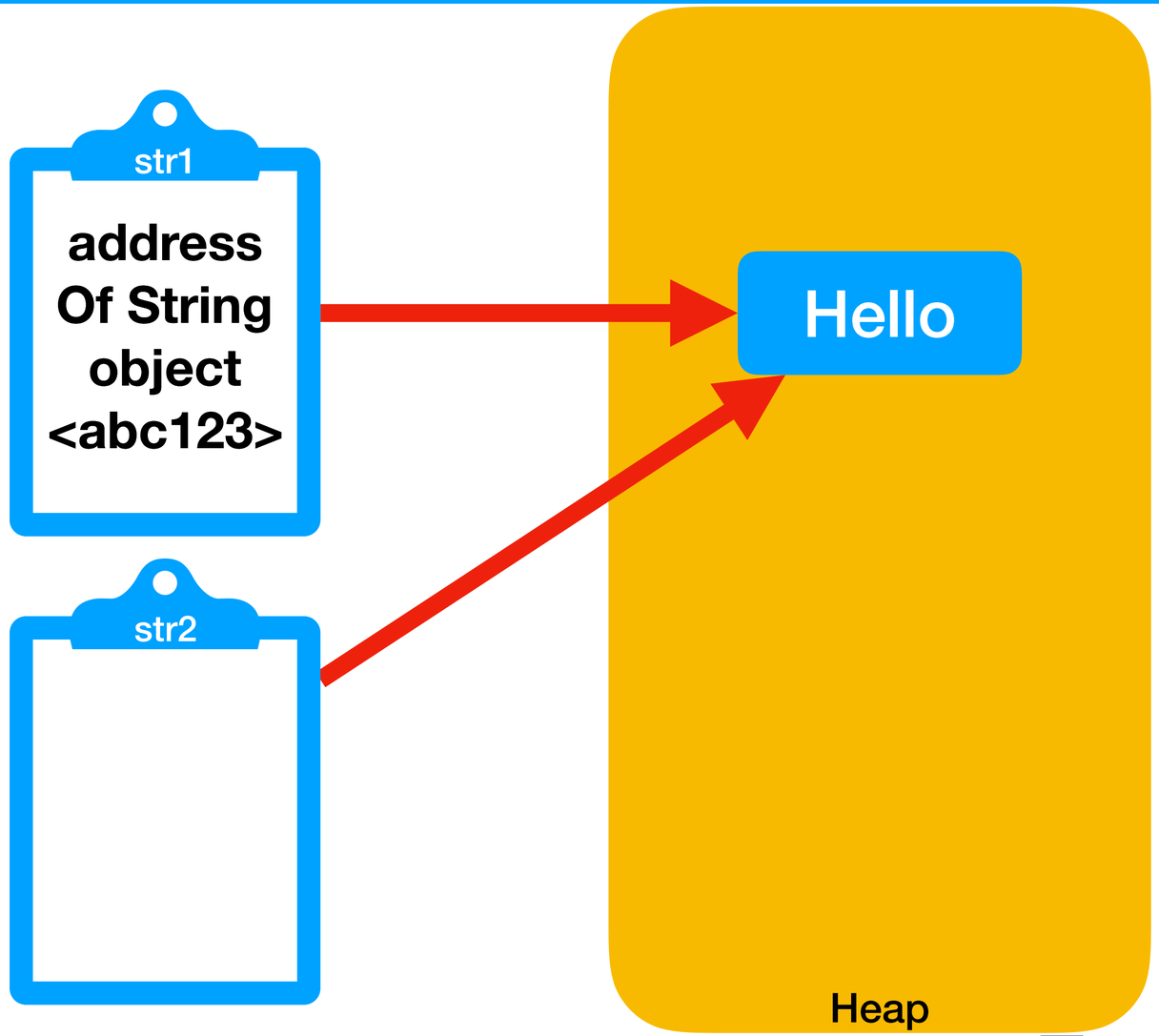

```
String str1 = new String("Hello") ;
String str2 = new String("Hello") ;
print(s1 == s2 ) ; //—> false
```

Cybertek

Equality recap with == for Reference type

```
String str1 = new String("Hello") ;
String str2 = str1;
print(str1==str2)  —> true
```

Both reference variable str1 and str2 has same address which point to same object. That's why == return true



Cybertek

String (constant) Pool

A special location in the heap to store unique string literals.

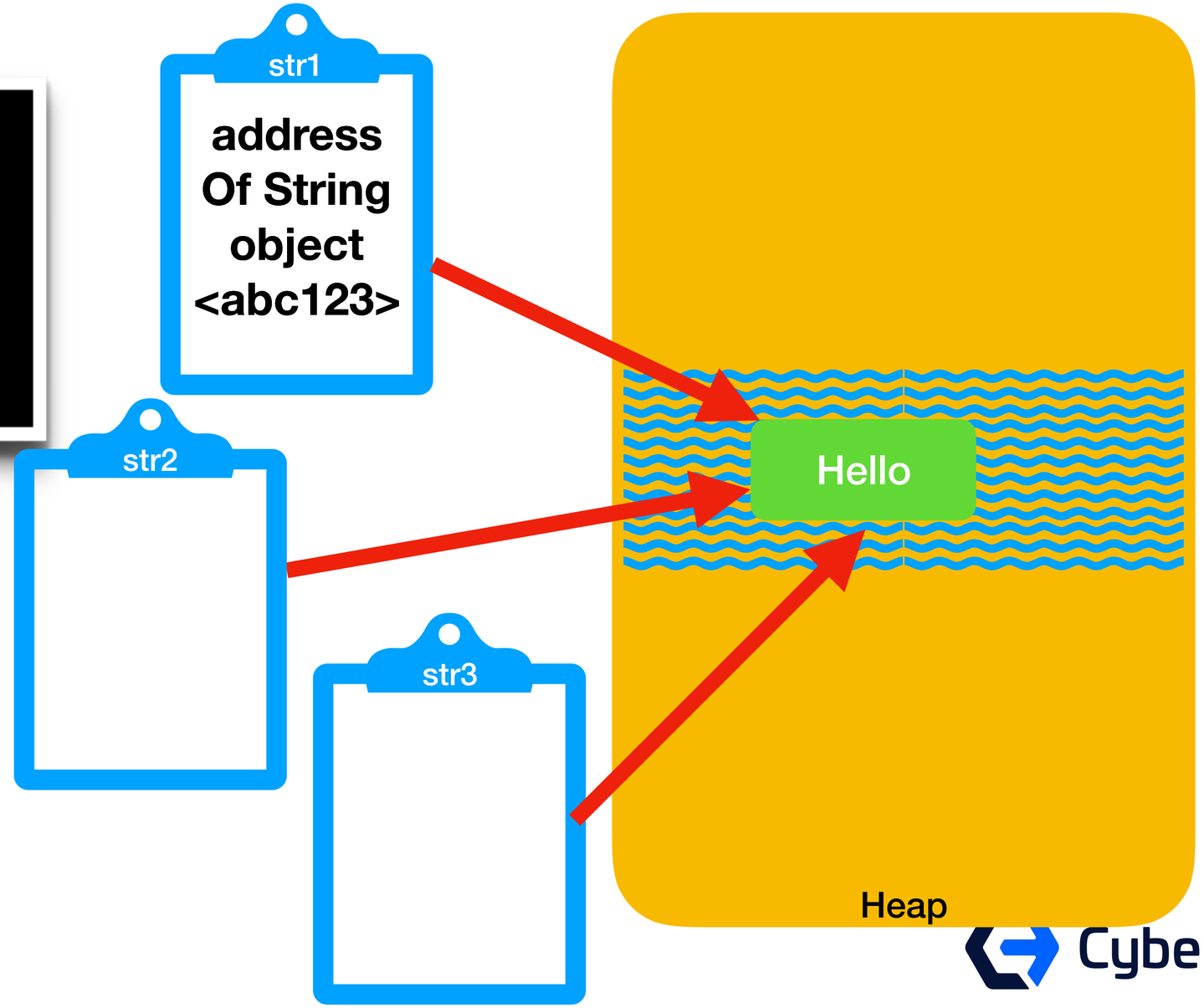
The ones created just by using “”
It helps to efficiently store strings without wasting memory

Cybertek

String Pool : A special location in the heap to store unique string literals.

```
String str1 = ”Hello” ;
String str2 = ”Hello” ;
String str3 = ”Hello” ;
print(str1==str2) —> true
print(str2==str3) —> true
```

- All reference variables have same address which point to same object.
- That’s why == return true

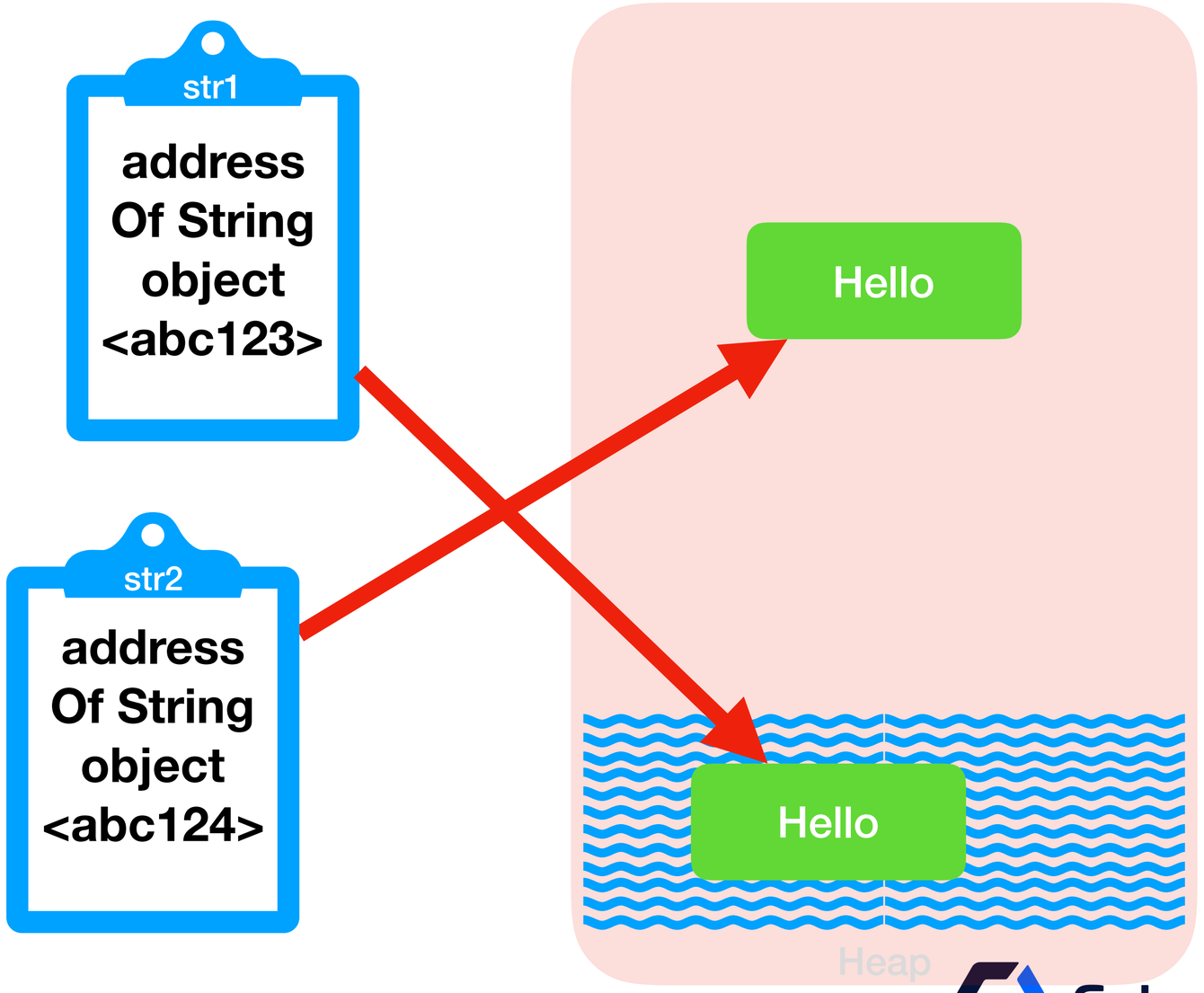


Cybertek

String Pool

```
String str1 = ”Hello” ;
String str2 = new String(”Hello”) ;
print(str1==str2) —> false
```

- Reference variable str1 and str2 has different address which point to different object.
- That’s why == return false



Cybertek

String Literals

A String created only by using “”

Not by new keyword , or method call or **variable** concatenation

in pool

```
String str1 = ”Hello” ;
String str2 = "Hel"+"lo";
print(str1==str2) —> true

String str = “lo” ;
```

Not in pool

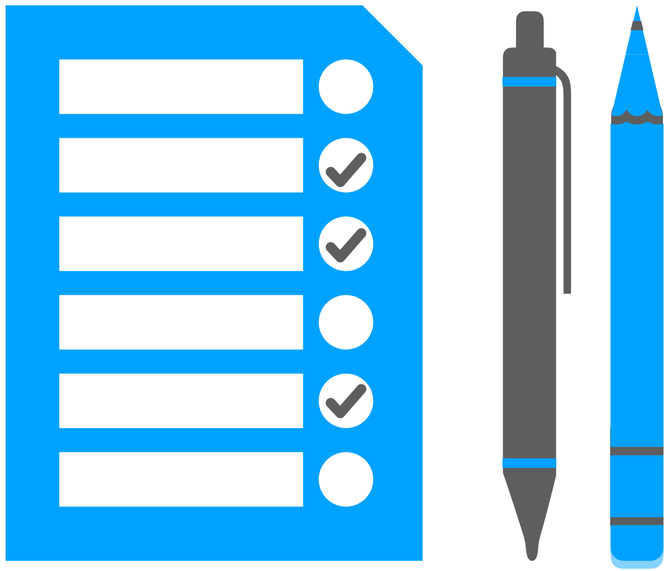
```
String str3 = new String(”Hello”)
String str4 = “Hel” + str;
String str5 = scan.next(); //hello

print(str3==str4)  —> false
print(str1==str3)  —> false
print(str1==str3)  —> false
print(str4==str5)  —> false
```

String Equality recap

.equal methods always should be used for checking 2 string have same content

== will only return true when two reference variables are pointing to same object.

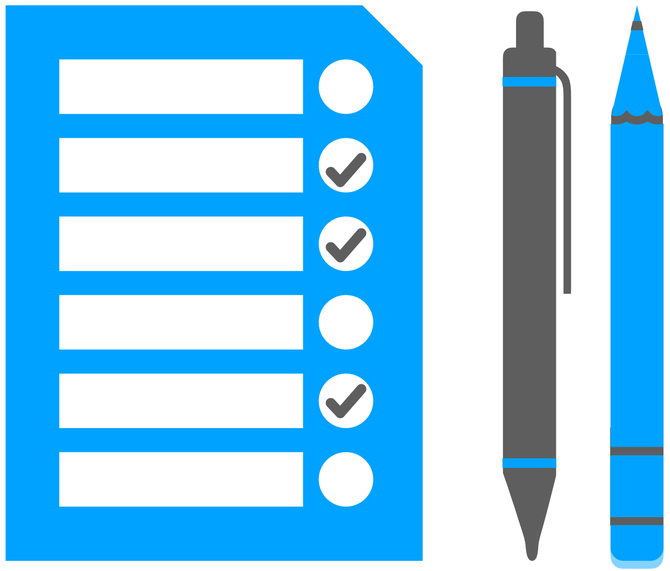
Loops



- Understand the concept of iteration
- Different Types of loops
  - While loop
  - Do while loop
  - For loop

Cybertek

After today’s session you should be able to:



- Understand the purpose of loops in programming
- Use loops to solve algorithm problems
- Write a simple program using while loop

Cybertek

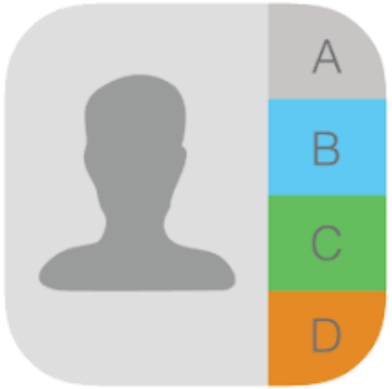
What is Iteration/Loop

Repetition / Looping

Looping allows you to execute one or more lines of code as many times as you need to

Loops repeats actions so you don’t have to

Repetition / Looping



Happy new Year
Happy new Year
Happy new Year
Happy new Year
Happy new Year
Happy new Year
Happy new Year
Happy new Year
Happy new Year
Happy new Year
Happy new Year
Happy new Year
Happy new Year
Happy new Year
Happy new Year
Happy new Year

Cybertek

Types of loop

While Loop

do while Loop

for loop

Cybertek

While Loop

```
//psudo code
while(A CONDITION IS TRUE ) {
  // REPEAT THIS ACTION
 }
```

While Loop

```
 //psudo code
while(there is still friend in contact ) {
  // Send Happy New Year Message
 }
```

Cybertek

While Loop

```
int x = 0 ;
while(x<5) {

 System.out.println("Happy new year");

}
```

CODE WILL KEEP REPEATING FOREVER UNTIL CONDITION IS FALSE

Cybertek

Without Loop

```
System.out.println("Happy new year");
System.out.println("Happy new year");
System.out.println("Happy new year");
System.out.println("Happy new year");
System.out.println("Happy new year");
```

Cybertek

While Loop

```
int x = 0 ;
while(x<5) {
 System.out.println("Happy new year");
 ++x ;
}
```

```
Happy new year
Happy new year
Happy new year
Happy new year
Happy new year
```

CODE WILL KEEP REPEATING FOREVER UNTIL CONDITION IS FALSE

In this case it will stop when x is 5 because 5<5 is false

Cybertek

While Loop

```
int x = 0 ;
while(x<5) {
 System.out.println("Happy new year");
 ++x ;
}
```

```
Happy new year
Happy new year
Happy new year
Happy new year
Happy new year
```

Cybertek