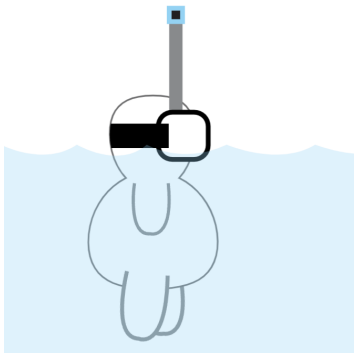
Free Diving

app design



occupation: diver

name: Jack Dawson

nationality: Australian

age: 25

Scenario

In a computer game, a free diver dives to catch fish.

The diver makes **10 dives**.

At the start of each dive, the diver has no fish and 100% air in lungs.

While the air in the diver's lungs is greater than 20%, the diver attempts to catch fish.

When the air supply is 20% or less, the diver returns to the boat, displays the catch data and dives again.

Challenges

Today's challenge will introduce another fundamental concept in computer program design - functions that return values.

It can be tricky. So - keep thinking and keep communicating with your partner (if you have one). You can work it out!

Getting started

Make sure you fully understand the problem (stated above) before you start to design the solution!!

Getting Started

Here is some code to help us get started.

```
public static void main(String[] args) {

    diveTrip();

}


public static void diveTrip() {

    //declare local variables
    int airSupply;
    int fishCaught;
    int totalFish;


    //this is the total fish of all 10 dives
    totalFish = 0;

    //10 dives
    for(int i = 0; i<10; i++) {

        airSupply = 100;
        fishCaught = 0;


        //attempt to catch fish while air suppy > 20
        while(airSupply>20) {

            //catch fish

            boolean attempt = catchFish();

            //if fish caught, yell() and increment the fishCaught variable
            if(attempt){
                yell();
                fishCaught = fishCaught + 1;
            }

            //reduce the airsupply!
            airSupply = reduceAirSupply(airSupply);

        }//end of current dive

        //update totalFish
        totalFish = totalFish + fishCaught;
    }

}//end of diveTrip
```

As you can see, there are 3 functions that need to be designed:

```
yell()
catchfish()
reduceAirSupply(int)
```

Let's use this function to produce a random yell... here is a silly idea:

```
public static void yell(){
    int rand = (int)(Math.random()*3);
    if(rand==0){
        System.out.println("                    ");
    }else if(rand==1){
        System.out.println("                    ");
    }else if(rand==2){
        System.out.println("                    ");
    }else{
        System.out.println("                    ");
    }

}
```

Can you see how it works? Design 4 messages that a diver might yell if (s)he catches a fish!

Now look at the call to catchFish(). It looks interesting?

```
boolean attempt = catchFish();
```

What is going on?! The program calls the function `catchFish()` and the function returns a value. That value is boolean! And the program that called the function has a variable ready to grab it, `attempt`.
So, the design of the catchFish() function will look like this::

```
public static boolean catchFish() {
    return true;
}
```

Can you see that we are not using `void` for this function. `void` means that the function does not return anything. But this function returns a `boolean`! So, we change `void` to `boolean`...
The function is not well-designed. We don't want to always return true. Can you redesign the `catchFish()` function so that it randomly returns `true` or `false`:

```
public static boolean catchFish() {

}
```

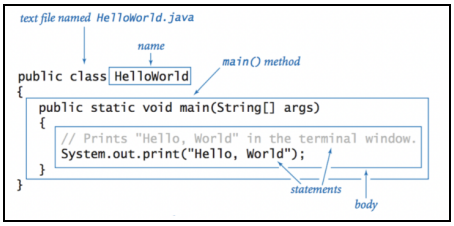
`reduceAirSupply()` takes an integer as a parameter. This is the current air supply. Somehow, it makes this number smaller and returns the result! This value is used to update the `airSupply` variable! Genius! Can you design the `reduceAirSupply` function:

```
public static ______ reduceAirSupply(int supply){

}
```

Code your design. Does it work? You may need to add some System.out statements. *The app design company wants there to be a mermaid and some sharks in the game. Can you add these adaptations using your own human creativity and ingenuity? Design functions that take parameters and/or return values!*

JAVA CHEAT SHEET



type	set of values	common operators	sample literal values
int	integers	+ - * / %	99 12 2147483647
double	floating-point numbers	+ - * /	3.14 2.5 6.022e23
boolean	boolean values	&& || !	true false
char	characters		'A' '1' '%' '\n'
String	sequences of characters	+	"AB" "Hello" "2.5"

Printing.

`void System.out.print(String s)`	*print s*
`void System.out.println(String s)`	*print s, followed by a newline*
`void System.out.println()`	*print a newline*

Nested if-else statement.

```
if      (income <      0) rate = 0.00;
else if (income <   8925) rate = 0.10;
else if (income <  36250) rate = 0.15;
else if (income <  87850) rate = 0.23;
else if (income < 183250) rate = 0.28;
else if (income < 398350) rate = 0.33;
else if (income < 400000) rate = 0.35;
else                      rate = 0.396;
```

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

Double num1 = scan.nextDouble();

Int num2 = scan.nextInt();

String word = scan.nextLine();
```

Comparison operators.

op	meaning	true	false
==	*equal*	2 == 2	2 == 3
!=	*not equal*	3 != 2	2 != 2
<	*less than*	2 < 13	2 < 2
<=	*less than or equal*	2 <= 2	3 <= 2
>	*greater than*	13 > 2	2 > 13
>=	*greater than or equal*	3 >= 2	2 >= 3

```
int rand = (int)(Math.random()*100);
```

Booleans.

values	*true or false*		
literals	true	false	
operations	and	or	not
operators	&&	||	!