Classes and Objects

Lecture 11 - Spring 2020

Follow-along #0: Construct a Pizza Object

- Before you begin coding, let's open Pizza.ts

- In 00-pizza-price-app.ts
  - Notice the Pizza class is imported from "./Pizza"
  - Refer to your notes / video slides for specific syntax

  1. Declare a variable and assign it a Pizza object. Print this object.

  2. Assign different values to each of its three properties (size, extraCheese, toppings). After doing so, print the object again.

  3. (Ignore Todo #3)

- Check-in on PollEv.com/compunc once complete!

```
// 1. Initialize a variable that holds a Pizza object and print it
let aPizza = new Pizza();
print(aPizza);

// 2. Assign different values to each of its properties
aPizza.size = "small";
aPizza.extraCheese = true;
aPizza.toppings = 2;
print(aPizza);
```

Hands-on #1: Calculate the Price of a Pizza

- Before you begin coding, open PizzaUtils.ts
  - Talk with your neighbor about what is inside of this file
- In 00-pizza-price-app.ts
  - Notice the **`price`** function is imported from "./PizzaUtils"

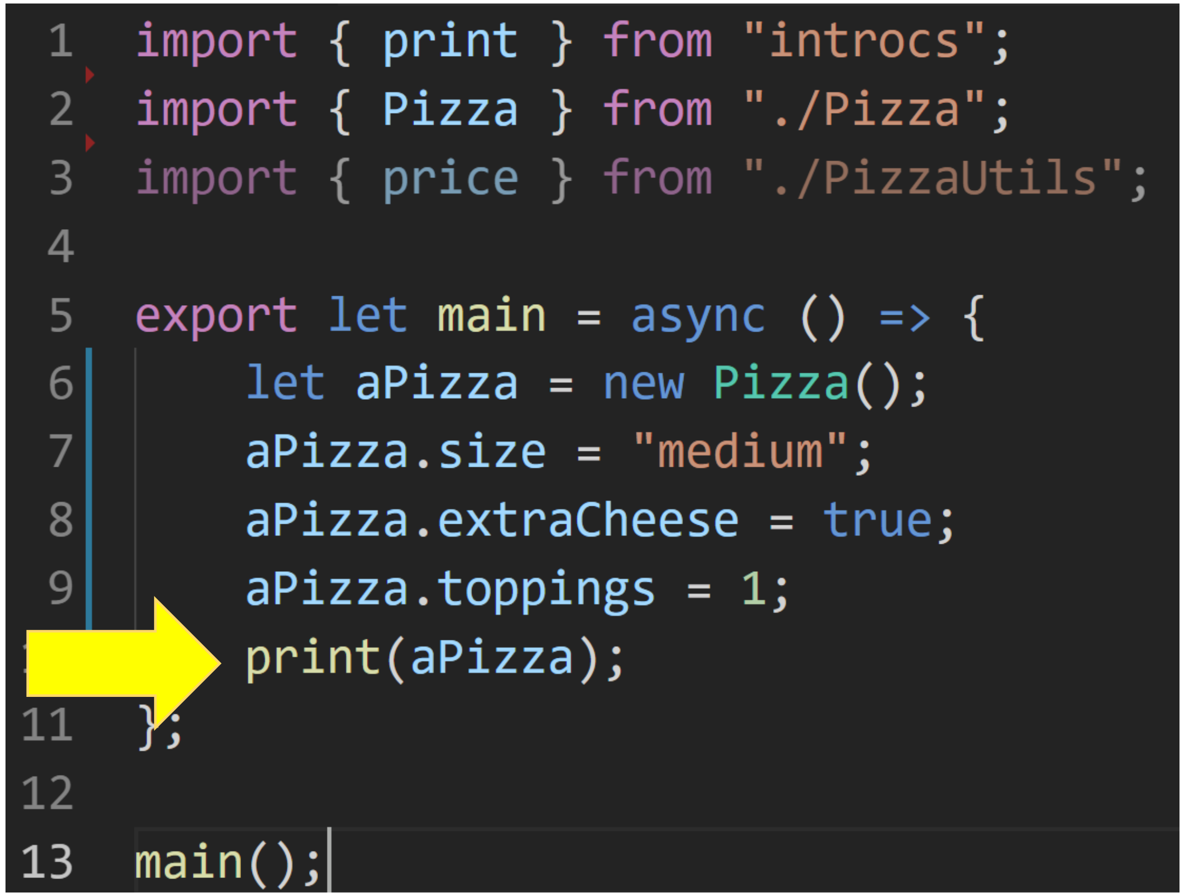
3. Call the **`price`** function with your Pizza object and print the return value. It should print 0 at this point because price is a skeleton function.

4. Correctly implement the **`price`** function in **PizzaUtils.ts**:
   - Size sets a base price of $7 small, $9 medium, $11 large
   - Extra cheese adds $1
   - Each topping costs $0.75

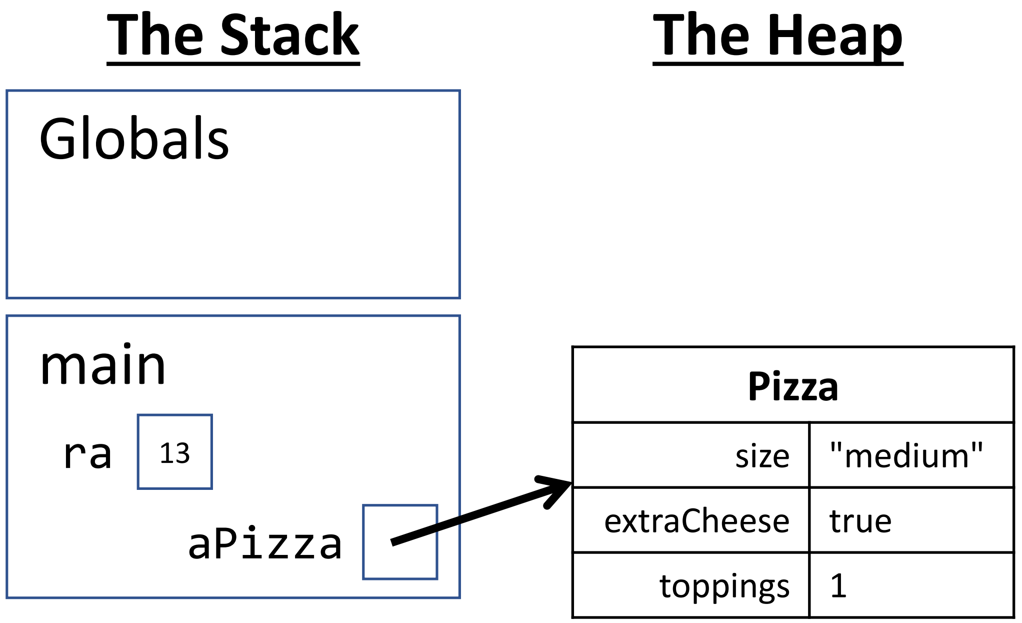
- Check-in on PollEv.com/compunc once your pizza price is correctly calculating! Try changing property values to inspect.

Object Values Live on the Heap

Like arrays, objects are *reference types*. Their variable names on the call stack hold references to their *actual values* in the heap.



```
import { print } from "introcs";
import { Pizza } from "./Pizza";
import { price } from "./PizzaUtils";

export let main = async () => {
    let aPizza = new Pizza();
    aPizza.size = "medium";
    aPizza.extraCheese = true;
    aPizza.toppings = 1;
    print(aPizza);
};

main();
```



Be Careful to **Always Initialize your Variables**

Common Error:

```
Uncaught TypeError: Cannot set property '<property>' of
undefined
```

- **Example:**

```
let pizza1: Pizza;
pizza1.size = "large"; // ERROR!!!
```

- **The fix:** `let pizza1 = new Pizza(); // Always initialize!`

```
// 3. Compute and print its price with the imported price function
print("The price is...");
print(price(aPizza));
```

```
export let price = (pizza: Pizza): number => {
    let cost = 0;
    if (pizza.size === "small") {
        cost = 7;
    } else if (pizza.size === "medium") {
        cost = 9;
    } else if (pizza.size === "large") {
        cost = 11;
    }

    if (pizza.extraCheese) {
        cost += 1;
    }

    cost += pizza.toppings * 0.75;

    return cost;
};
```

The "Bundling" of Related Values is an Important Benefit of Composite Data Types / Objects

- Consider the following two function signatures...

```
let price = (size: string, extraCheese: boolean, toppings: number): number => {}

let price = (pizza: Pizza): number => {};
```

- Notice with a Pizza data type the function's *semantics* are improved
  - Is the first function calculating the price of a cheeseburger?
  - The second function's signature reads more meaningfully...
    "`price` is a function that is given a `Pizza` object and returns a `number`"

- Consider an object with *far more* properties...
  - Pizza: Base sauce, gluten free crust, thin vs. deep dish, ...
  - Objects give us a convenient means for tightly packaging related variables together

Arrays of Objects

- You can make an array of objects!

  Declaration is just the same...

```
let <arrayName>: <type>[] = [];
ex: let orders: Pizza[] = [];
```

- Initializing an element requires constructing an object:

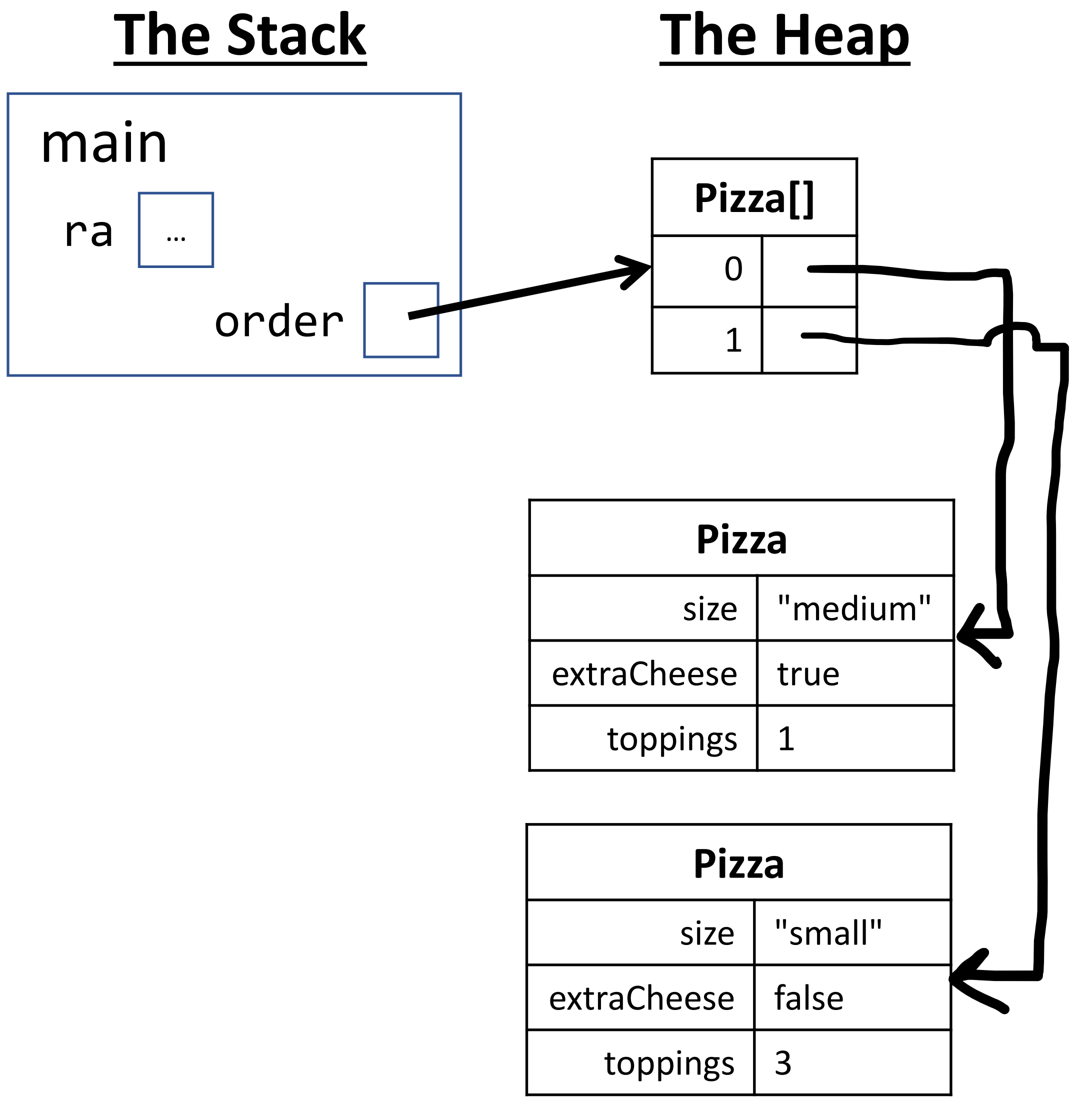
```
<arrayName>[<index>] = new <type>();
ex: orders[0] = new Pizza();
```

- Accessing an element is also the same:

```
<arrayName>[<index>]
ex: orders[1]
```

- To access a property, use the dot operator:

```
<arrayName>[<index>].<propertyName>
ex: orders[1].toppings
```



Follow Along: Working with Arrays of Objects

- Open 01-pizza-order-app.ts
- Notice that the **`order`** variable's type is a **`Pizza[]`**
- After the while loop completes:

1. Print the **`order`** array
2. Print the first element of the **`order`** array
3. Print the **`size`** property of the first element of the **`order`** array

```
print("The order is...");
// TODO 1: Print the order
print(order);

print("The first Pizza is...");
// TODO 2: Print the 1st pizza at index 0
print(order[0]);

print("The first Pizza's toppings are...");
// TODO 3: Print the 1st pizza's toppings
print(order[0].toppings);
```

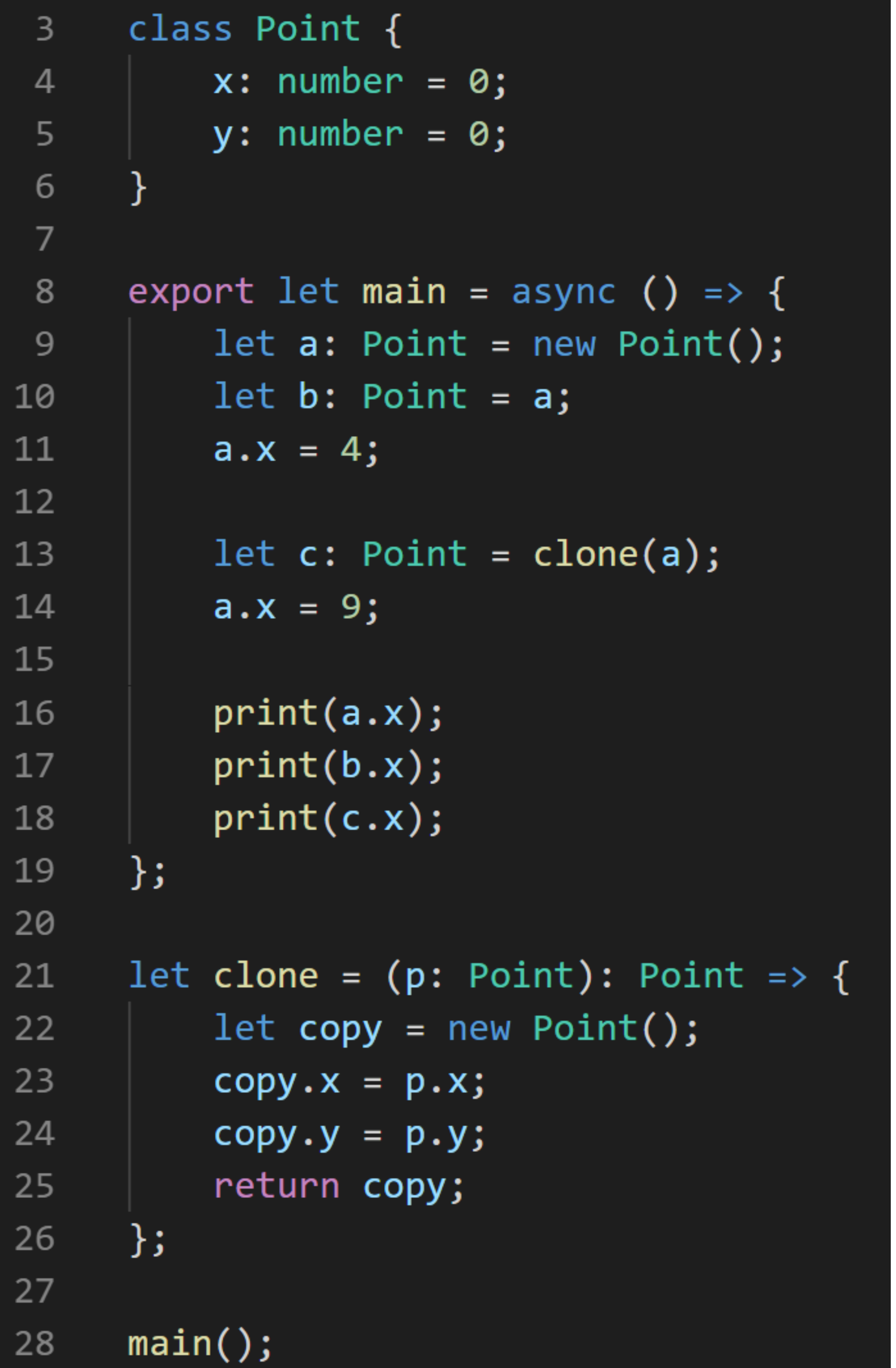
Hands-on: Iterating over an Array of Objects

- In 01-pizza-order-app.ts
- In the **`main`** function, call the **`orderPrice`** function and print its return value.
- Then, correctly implement the **`orderPrice`** function skeleton:

1. Loop over each of the Pizza objects in the pizzas parameter
2. Call the **`price`** function (imported) with each pizza
3. Add the price of each pizza to the total

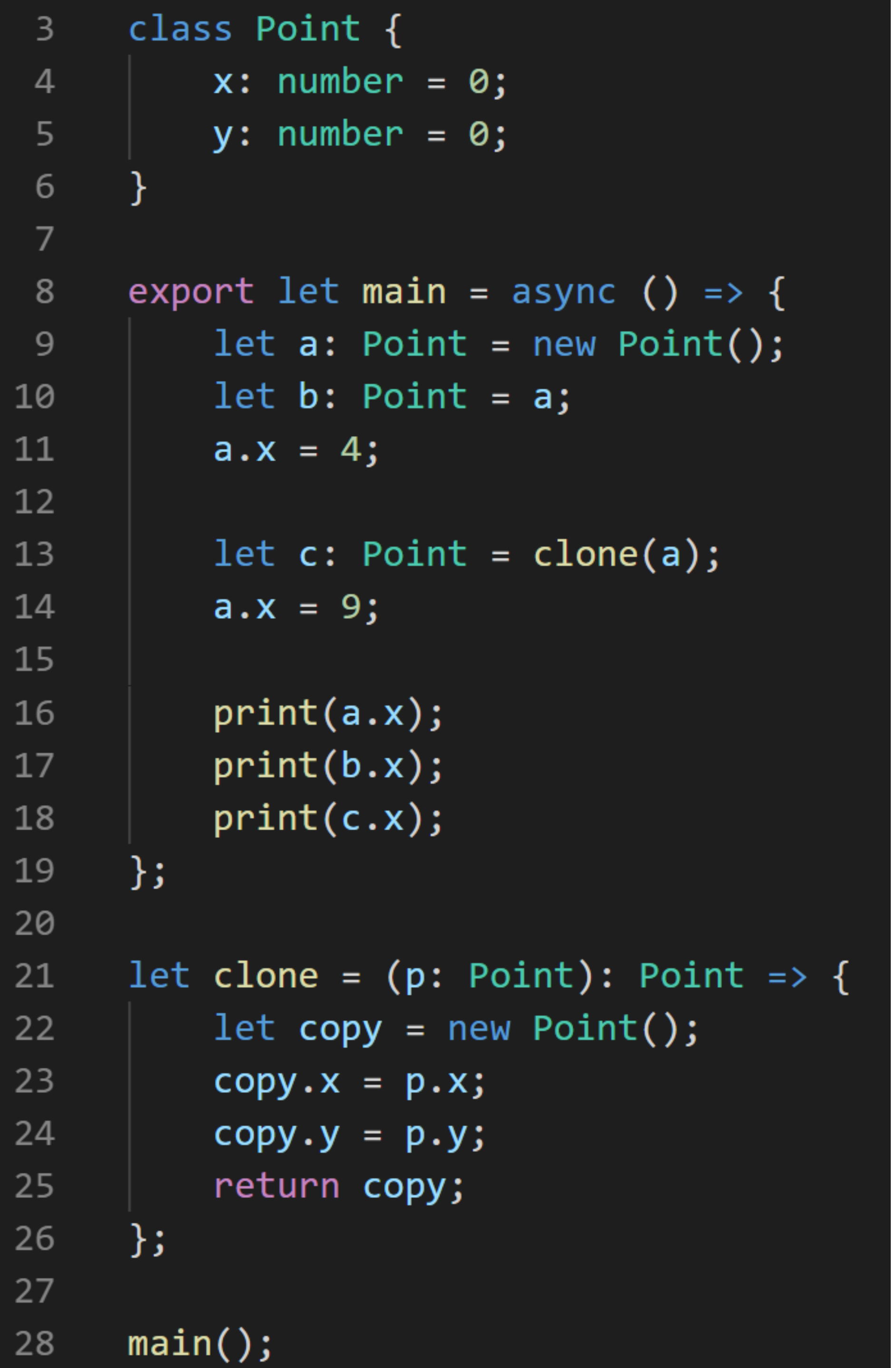
- Check-in when you're calculating the total price of an array of Pizzas.

```
let orderPrice = (pizzas: Pizza[]): number => {
    let total = 0;

    // TODO: Calculate the total price of an array of Pizzas
    for (let i = 0; i < pizzas.length; i++) {
        total += price(pizzas[i]);
    }

    return total;
};
```

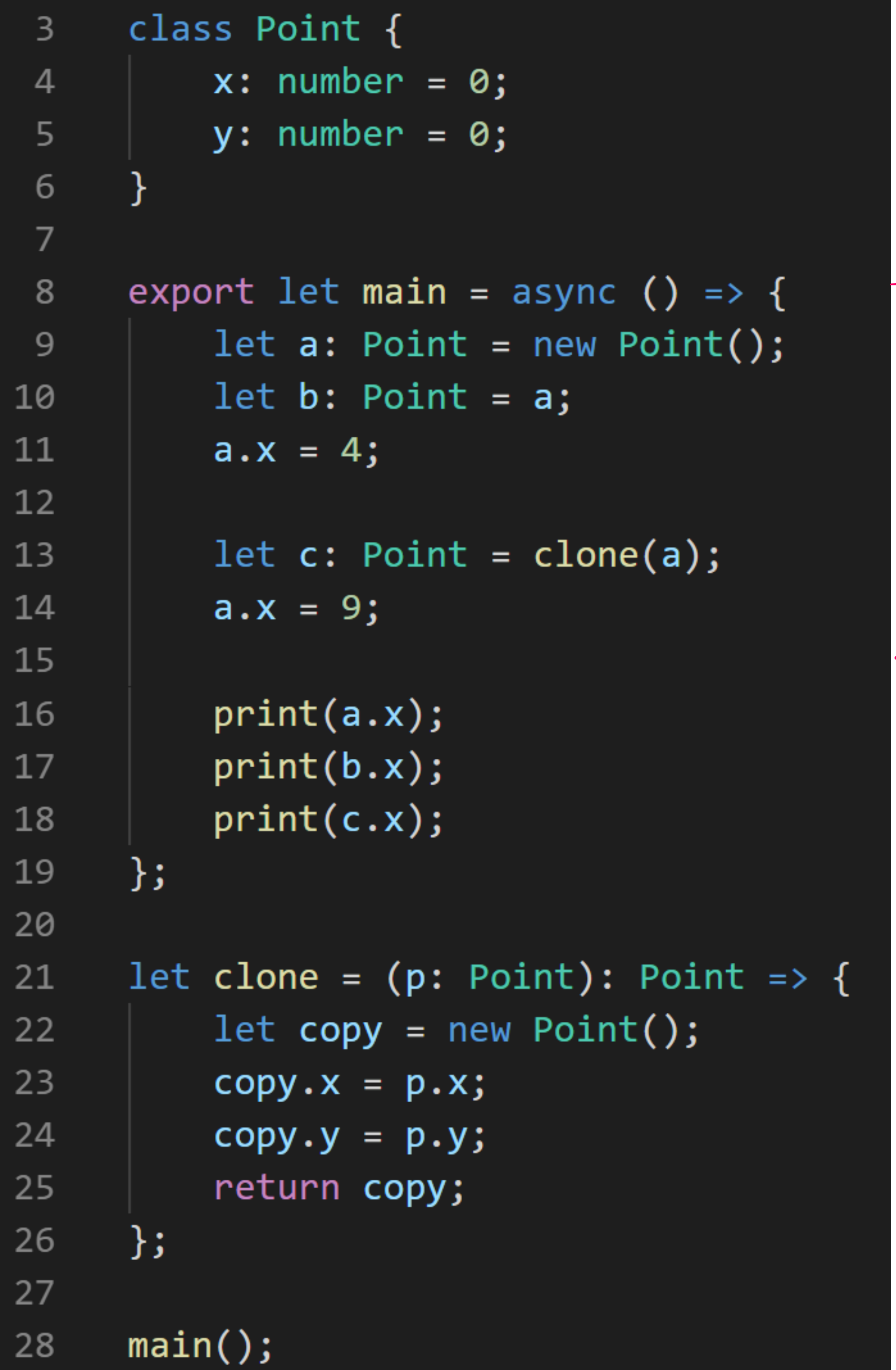


```typescript
class Point {
    x: number = 0;
    y: number = 0;
}

export let main = async () => {
    let a: Point = new Point();
    let b: Point = a;
    a.x = 4;

    let c: Point = clone(a);
    a.x = 9;

    print(a.x);
    print(b.x);
    print(c.x);
};

let clone = (p: Point): Point => {
    let copy = new Point();
    copy.x = p.x;
    copy.y = p.y;
    return copy;
};

main();
```

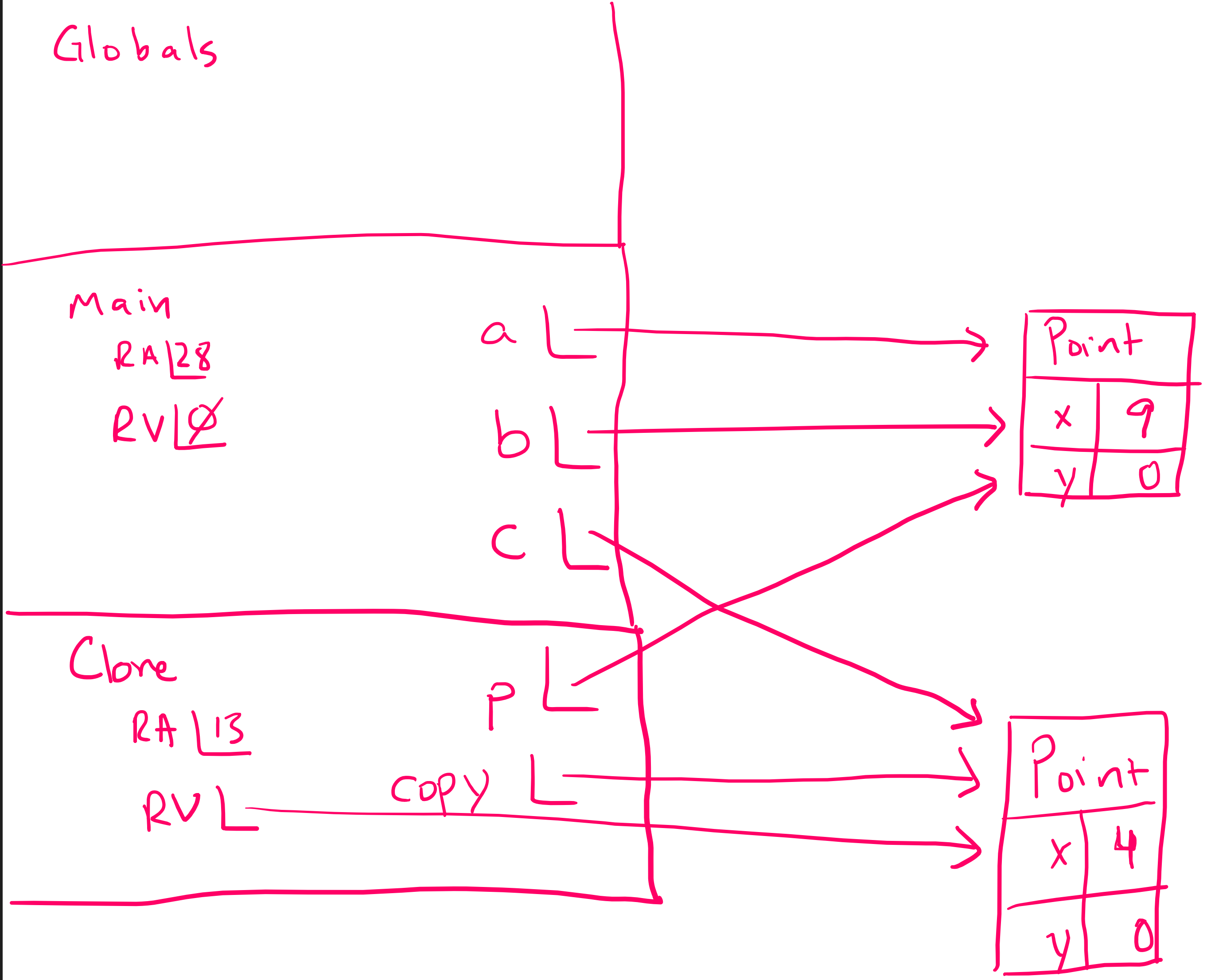
Challenge Question 1. Draw an environment diagram of the code listing and respond to the sequence of questions on PollEverywhere once completed.



```typescript
class Point {
    x: number = 0;
    y: number = 0;
}

export let main = async () => {
    let a: Point = new Point();
    let b: Point = a;
    a.x = 4;

    let c: Point = clone(a);
    a.x = 9;

    print(a.x);
    print(b.x);
    print(c.x);
};

let clone = (p: Point): Point => {
    let copy = new Point();
    copy.x = p.x;
    copy.y = p.y;
    return copy;
};

main();
```



```
3   class Point {
4       x: number = 0;
5       y: number = 0;
6   }
7
8   export let main = async () => {
9       let a: Point = new Point();
10      let b: Point = a;
11      a.x = 4;
12
13      let c: Point = clone(a);
14      a.x = 9;
15
16      print(a.x);
17      print(b.x);
18      print(c.x);
19  };
20
21  let clone = (p: Point): Point => {
22      let copy = new Point();
23      copy.x = p.x;
24      copy.y = p.y;
25      return copy;
26  };
27
28  main();
```



Output: 9
9
4