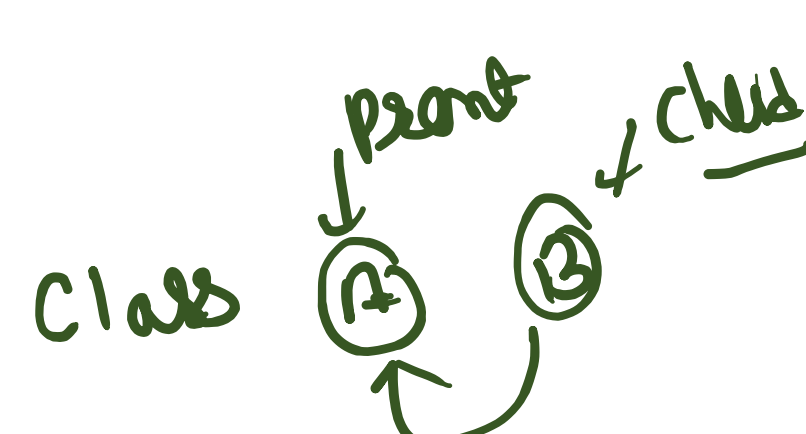
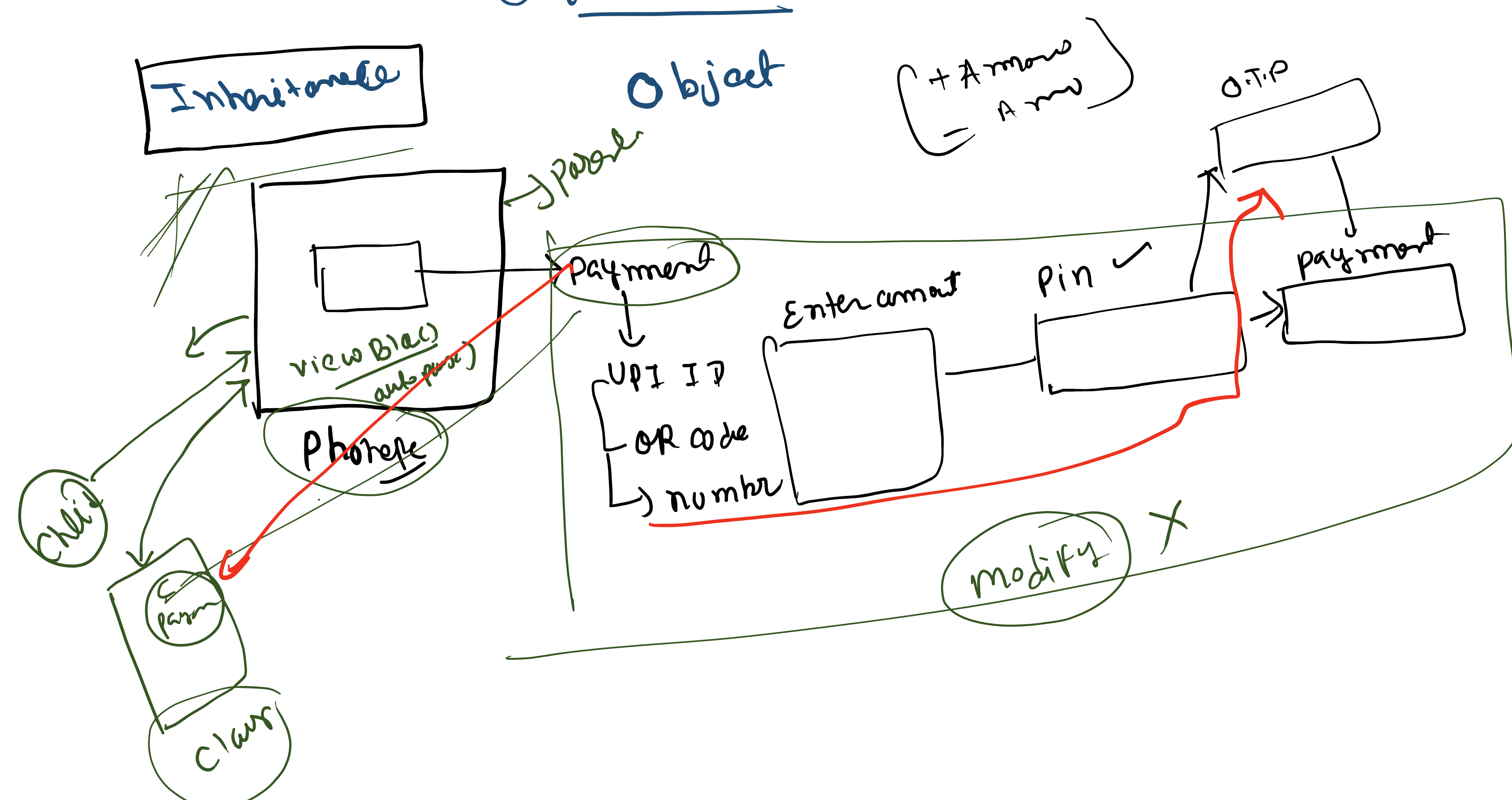
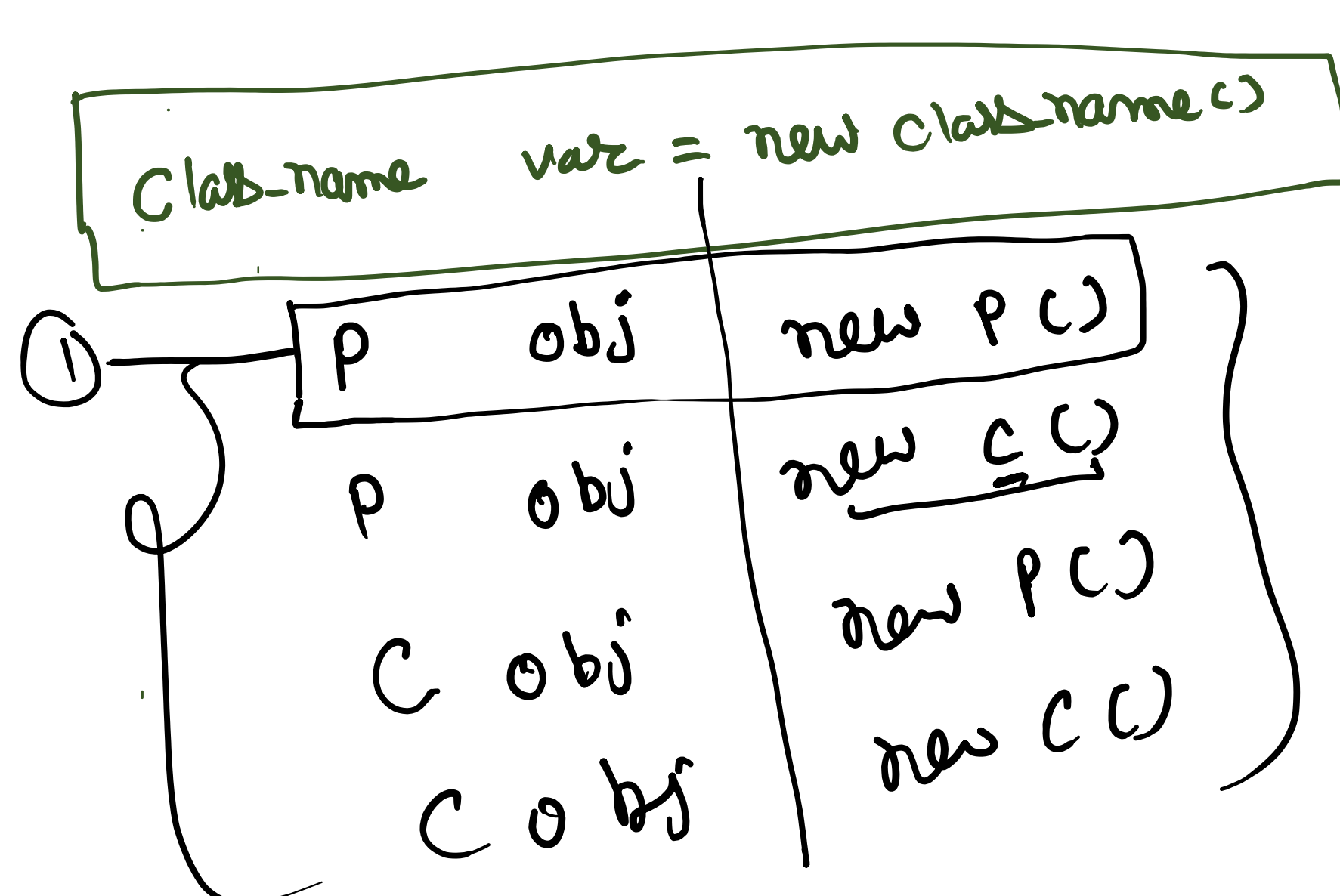


① toString()



$C \rightarrow P$



```

P obj = new P();
System.out.println(obj.d); // 1
System.out.println(obj.d1); // 10
obj.fun1();

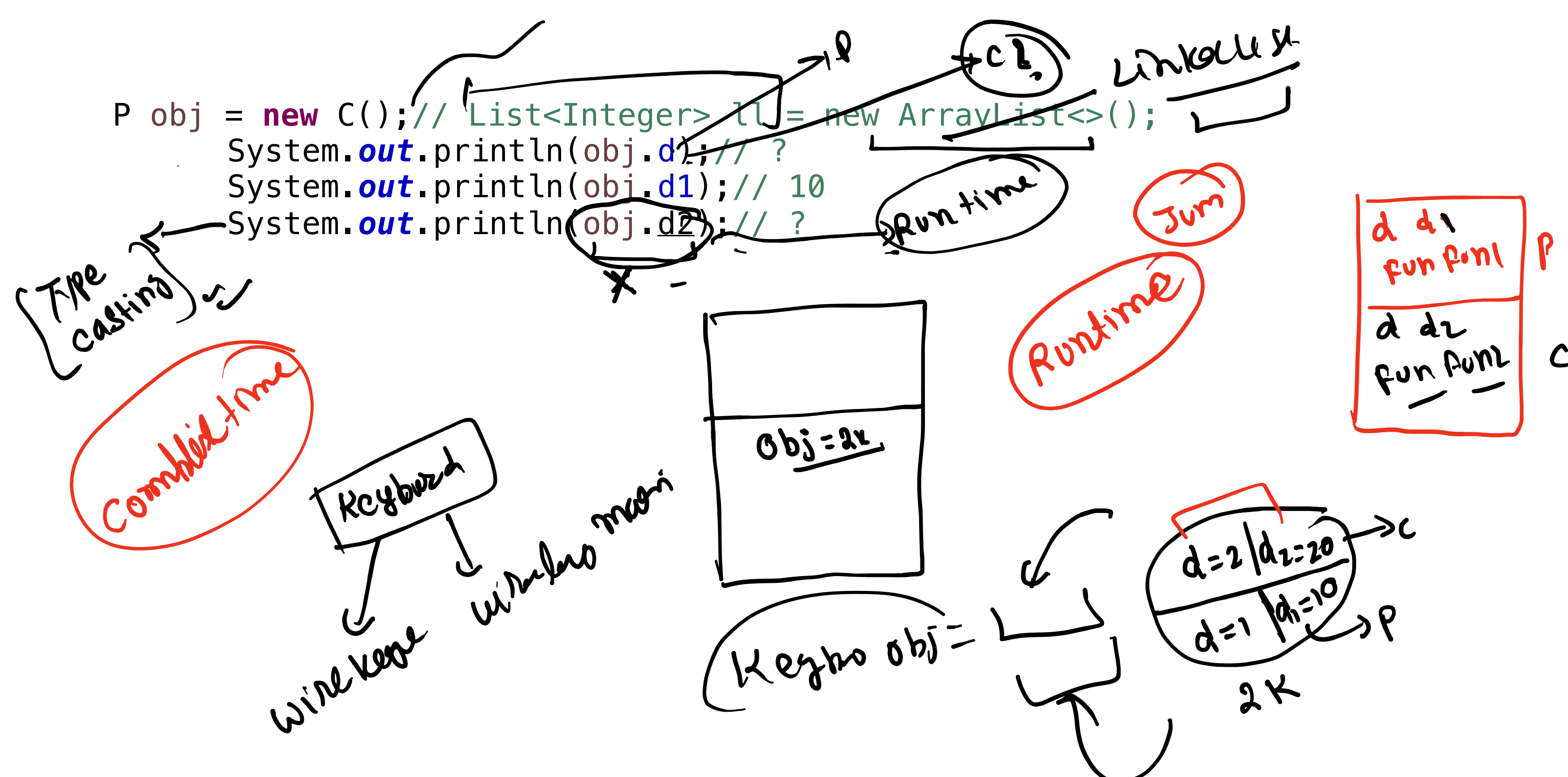
```

Handwritten annotations on the code:

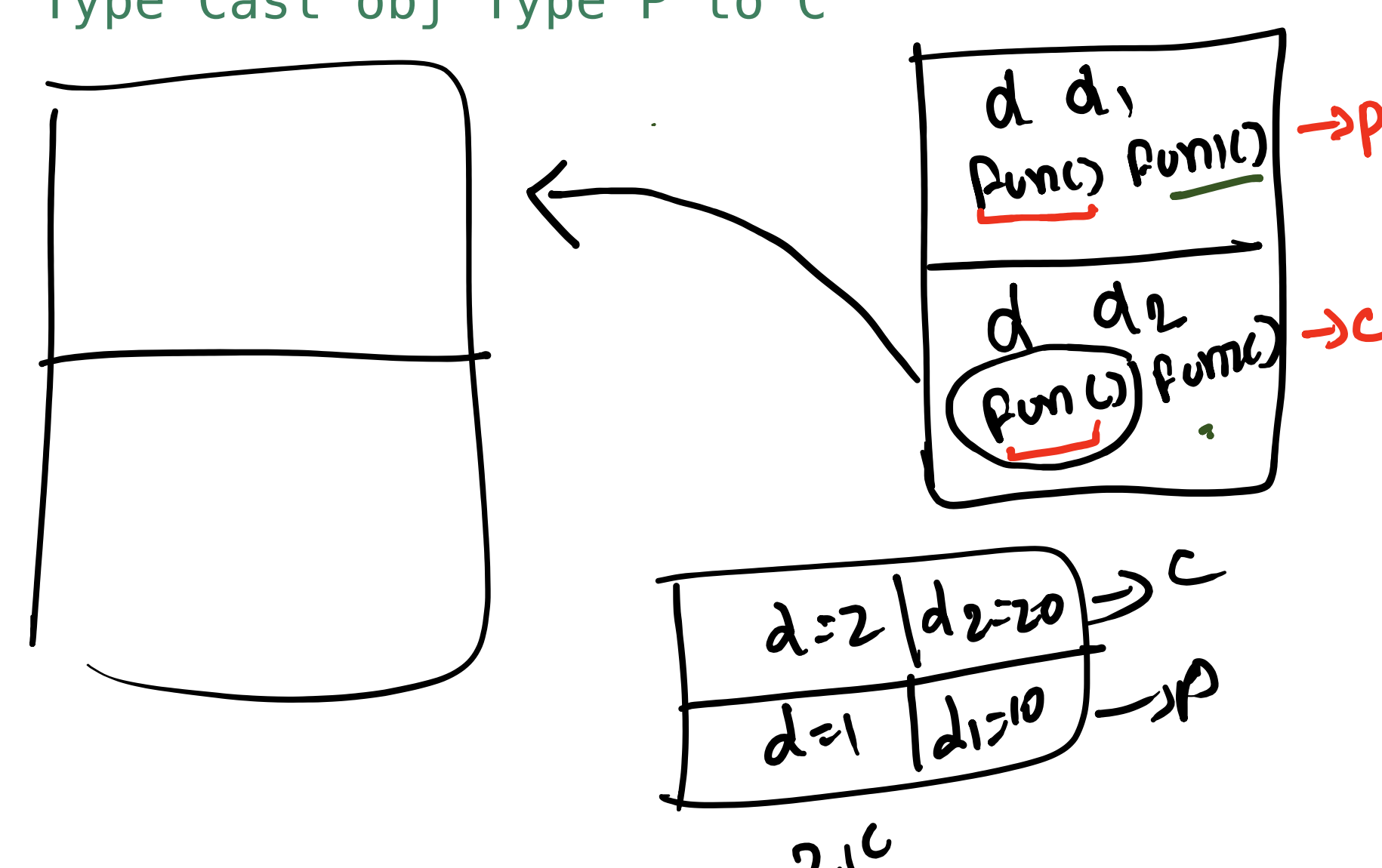
- class** (circled in green) points to `P`.
- variable** (written in green) points to `obj`.
- object create** (written in green) points to `new P()`.
- RHS** (written in green) points to `new P()`.
- Data Type** (written in green) points to `P`.
- LHS?** (written in green) points to `obj`.

```
P obj = new C(); // List<Integer> ll = new ArrayList<>();
System.out.println(obj.d1); // ?
System.out.println(obj.d1); // 10
System.out.println(obj.d2); // ?
```

Handwritten annotations:
 - A bracket above the first line points to the text "Unvollständig".
 - A bracket above the second line points to the text "Run time".
 - A bracket above the third line points to the text "Jum".



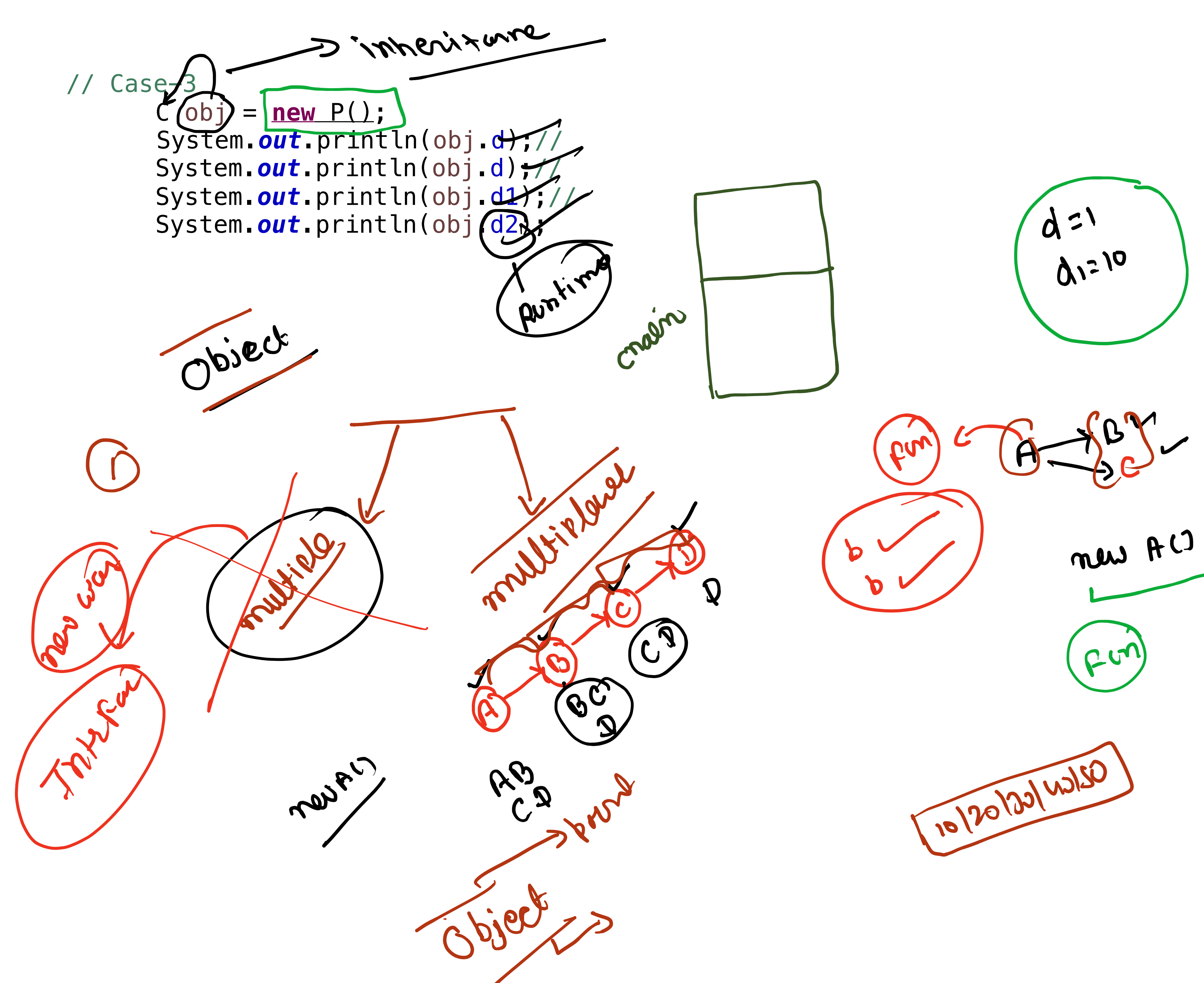
```
// Case-2
P obj = new C(); // List<Integer> ll = new ArrayList<>();
System.out.println(obj.d); // 1
System.out.println(((C) obj).d); // 2
System.out.println(obj.d1); // 10
System.out.println(((C) obj).d2); // Type Cast obj Type P to C
obj.fun1(); // C
obj.fun1(); // P
((C) obj).fun2();
```



// Case 3

```
obj = new P();
System.out.println(obj.d);
System.out.println(obj.d1);
System.out.println(obj.d1);
System.out.println(obj.d2);
```

inheritance

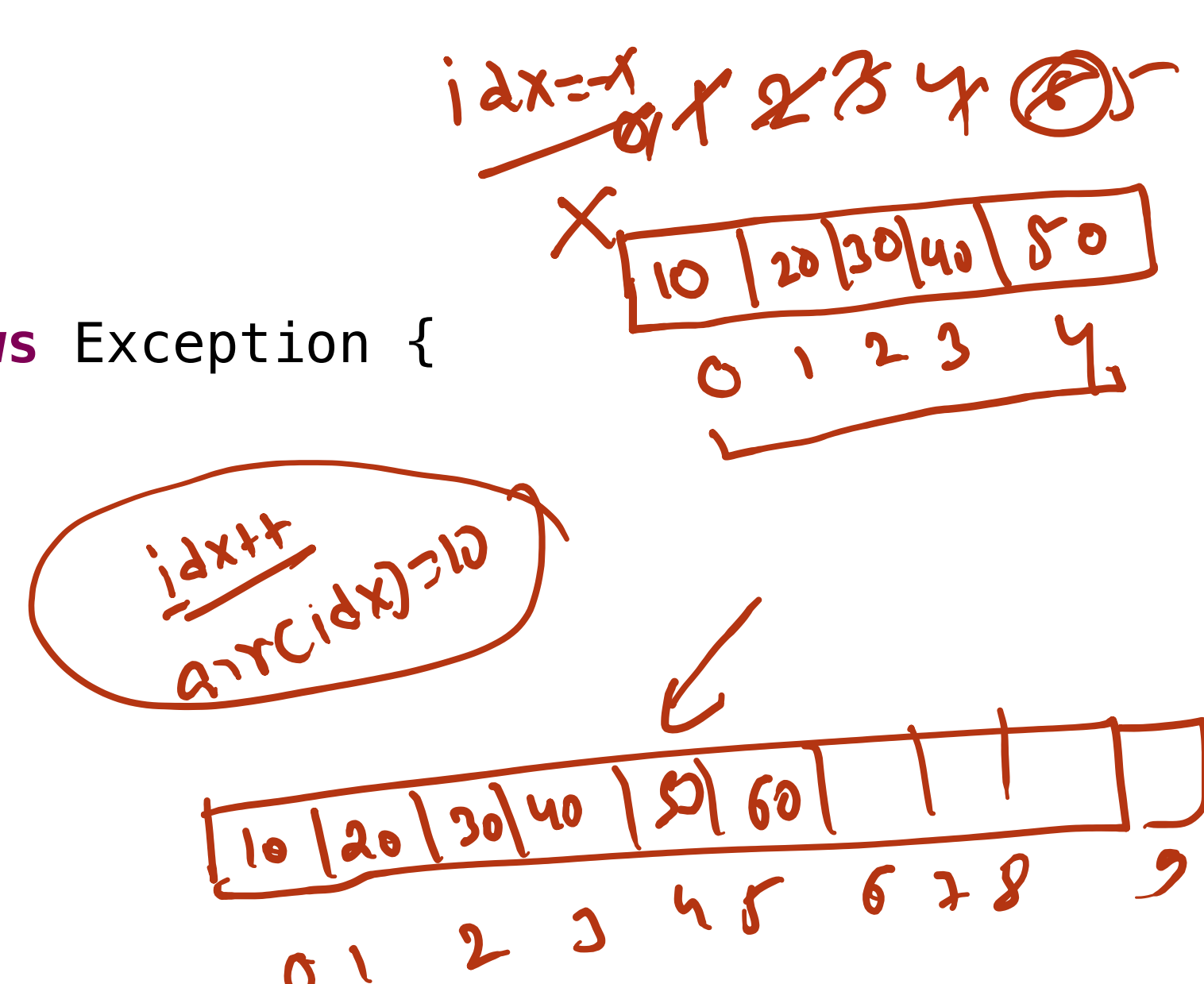


```
@Override
public void push(int item) {
}

public static void main(String[] args) throws Exception {
    Stack st = new DynamicStack(); // push
    // DynamicStack st1 = new DynamicStack();
    st.push(10); ✓
    st.push(20); ✓
    st.push(30);
    st.push(40);
    st.push(50);
    st.push(60); ✓
    st.push(70);
}
```

idx ↑
arr[idx]

10 | 20



The image shows two snippets of Java code with handwritten annotations.

Snippet 1:

```
public void push(int item) {  
    if (isFull()) {  
        int[] new_arr = new int[2 * arr.length];  
        for (int i = 0; i < arr.length; i++) {  
            new_arr[i] = arr[i];  
        }  
        arr = new_arr;  
    }  
}
```

Annotations for Snippet 1:

- A red bracket on the left side of the `if` block spans from `if (isFull())` down to `arr = new_arr;`.
- An arrow points from `new_arr` to its declaration `int[] new_arr = new int[2 * arr.length];`.
- An arrow points from `arr` to its assignment `arr = new_arr;`.
- A green note next to the assignment says `arr[idx++] = item`.
- To the right, there are two arrays:
 - Top array: `idx = 10 + 2 * 4` above it. The array contains `[10, 20, 30]`. Below it are indices `0 1 2`.
 - Bottom array: Contains `[10, 20, 30, 40, 50, 60, 70]`. Below it are indices `0 1 2 3 4 5 6 7`.

Snippet 2:

```
public static void main(String[] args) throws Exception {  
    Stack st = new DynamicStack(10); // push  
    // DynamicStack st1 = new DynamicStack();  
    st.push(10); ✓  
    st.push(20); ✓  
    st.push(30); ✓  
    st.push(40); ✓  
    st.push(50); ✓  
    st.push(60); ✓  
    st.push(70); ✓✓
```

Annotations for Snippet 2:

- Red checkmarks are placed after each successful `push` operation.
- A red checkmark is also placed at the end of the `main` method.

```
public void Enqueue(int item) {
    // TODO Auto-generated method stub
    if (isFull()) {
        super.Enqueue(item);
    }
}

public static void main(String[] args) {
    DynamicQueue dq = new DynamicQueue();
    dq.Enqueue(10);
    dq.Enqueue(20);
    dq.Enqueue(30);
    dq.Enqueue(40);
    System.out.println(dq.Dequeue());
    System.out.println(dq.Dequeue());
    dq.Enqueue(50);
    dq.Enqueue(60);
    dq.Enqueue(70);
    dq.Enqueue(80);
    dq.Display();
}
```

