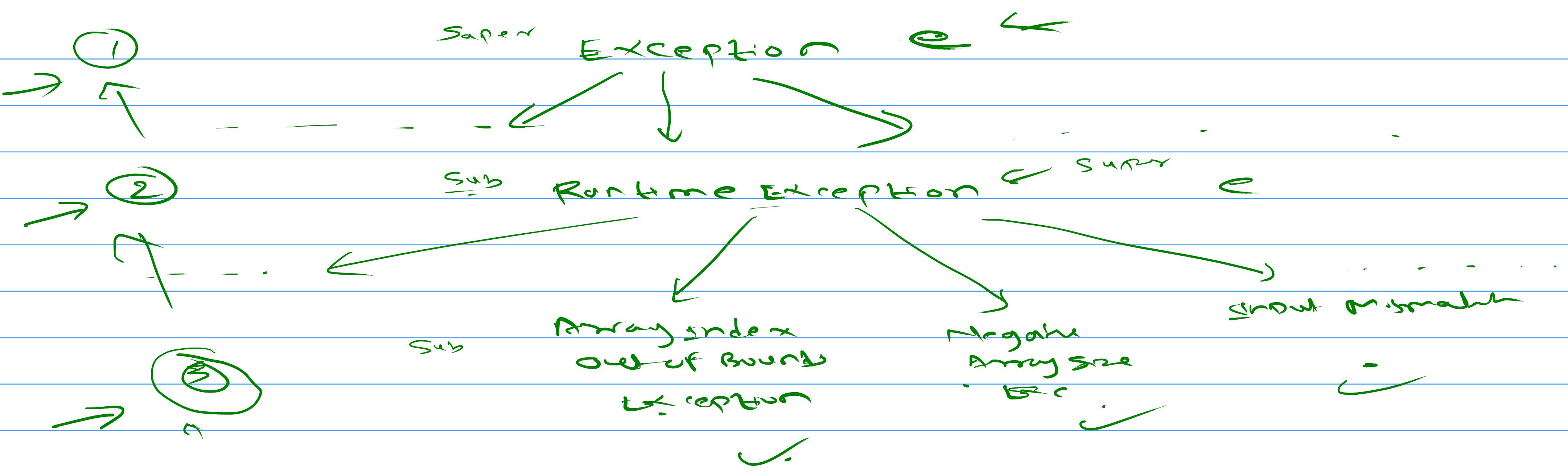
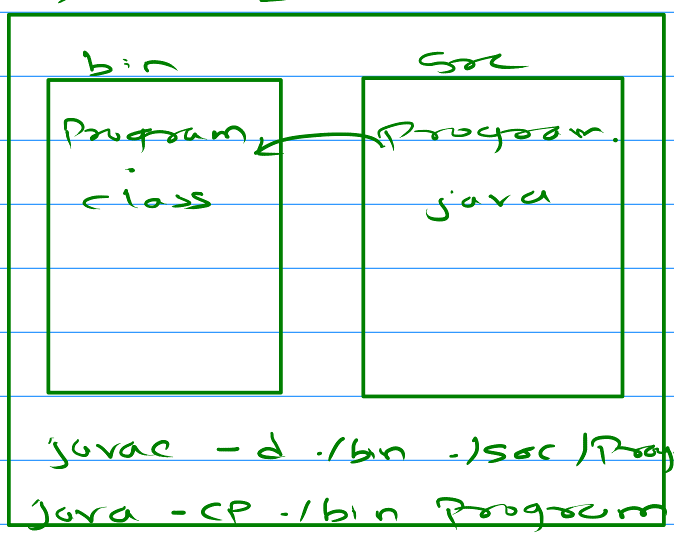

1
2
3
Super
Exception
e
Sub
Runtime Exception
super
e
Sub
Array index out of Bounds Exception
Negative Array size Exc
Input Mismatch

Demo 01



```
javac -d ./bin ./src/Program.java
java -cp ./bin Program
```

Demo 02

```
|
|--> src
|     └-> Program.java
|
|--> bin
      └-> P1
           └-> Program.java
```

Demo 03

```
|
|--> src
|     |--> Program.java
|     └--> Complex.java   (P1)
|
└--> bin
      └--> Program.class
           P1
             └--> Complex.class
```

Demo 04

```
|
|--> src
|     └-> P1
|     |    └-> Program .ja
|     └-> P2
|          └-> complex.java
|
└--> bin
      └-> P1
      |    └-> Program.class
      └-> P2
           └-> Complex.class
```

$$\begin{tikzcd}
 & \text{Hirerachy} \arrow[dl, "\text{has-a}"'] \arrow[dr, "\text{is-a} \rightarrow"] & \\
\text{Association} \arrow[d] \arrow[dr] & & \text{Inheritance} \\
\text{Aggregation} \arrow[d] & \text{Composition} \arrow[d] & \\
\text{loose Coupling} & \text{tight Coupling} &
\end{tikzcd}$$

```
Person                                  Date {
{
  Date ~~dob;~~
  Person( ,dob, ) ×                                  Person( )
  Person(      ) ×                                   ✓ new Date()

                                                     Person(int d, m, y) {
                                                     ✓new Date();
}                                       }            }

main( )                                              main( )
{                                                    {
 Date d = new Date();                                Person p =
 Person p = new Person();                              new Person();
 Person p = new Person(d);                           }

}
```

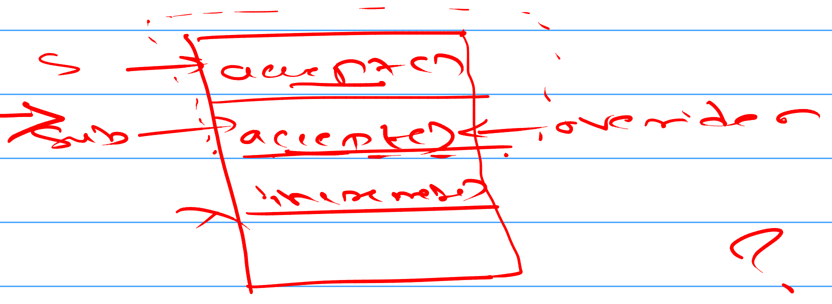
Dependency obj is created
outside the class
Aggregation

```
Super  ──→ class Person (Parent)
                 │
over Data( )     ↓
Sub    ──→ Class Employee (Child)
over.i Data( )
{
}    Super()   Sub class → object(   ,   );
```

object

new superParam

Super s = new Sub(); // upcasting



?

s.accept()

Sub sub1 = (Sub) s;

Emp emp =

Shape → abstract area() → 1,2,3,5

abstract calculate →

?   ?

abstract class Shape {
int num1;
Shape()
{
}
getArea() {
}
}

interface
{
}

?

class extends class   interface extends interface

implements interface

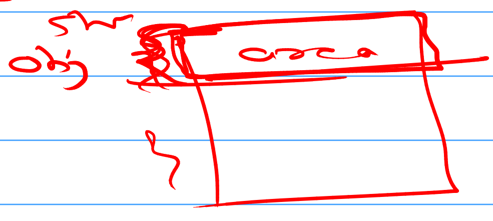
?

impl of

ext class

impl ca?

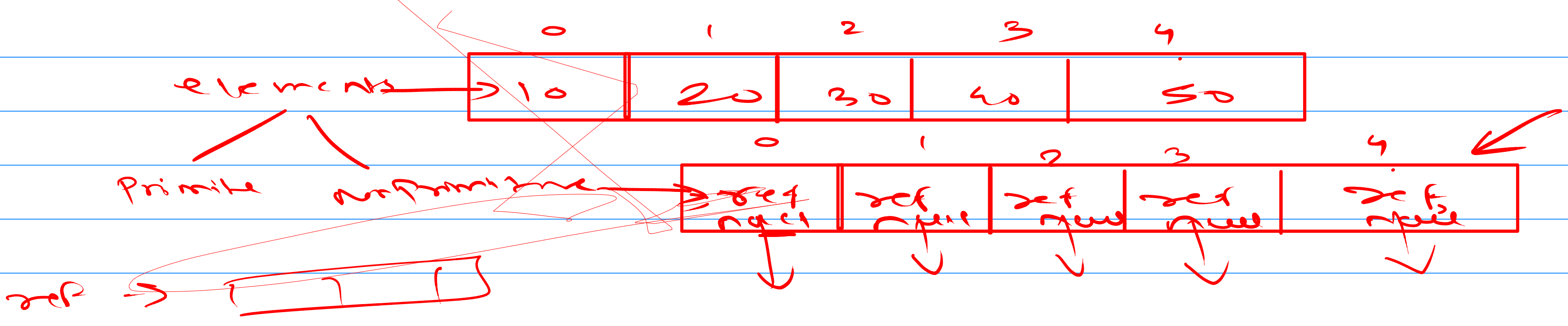
Shape s = ~~new Shape();~~ new Rectangle(); s
s = new Circle();



s instanceof R ?
C
Shape

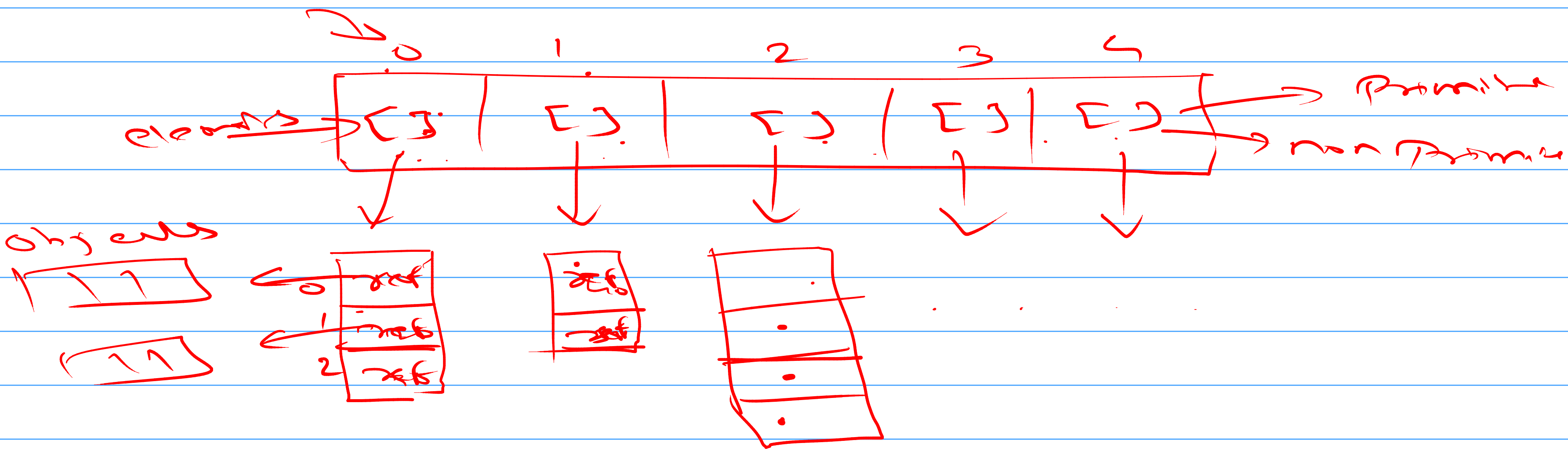
True

Rectangle r = new Rectangle → True
r instanceof Shape



```
int arr[] = new int[5]  Ex
    arr[0] = 10;
Emp arr[] = new Emp[5]
    arr[0] = new Emp();

for ( ; ; )
for ( ; )
```



```
int [][] arr = new int[5][];
    arr[0] = new int[2];
    arr [0] = new int [3];

    arr[0][0] = 10

Emp [][] arr = new Emp [3][ ];
    // arr[0] = new Emp [2];
    // arr[1] = new Emp [3];
for(      )
    arr [0][0] = new Emp();
```

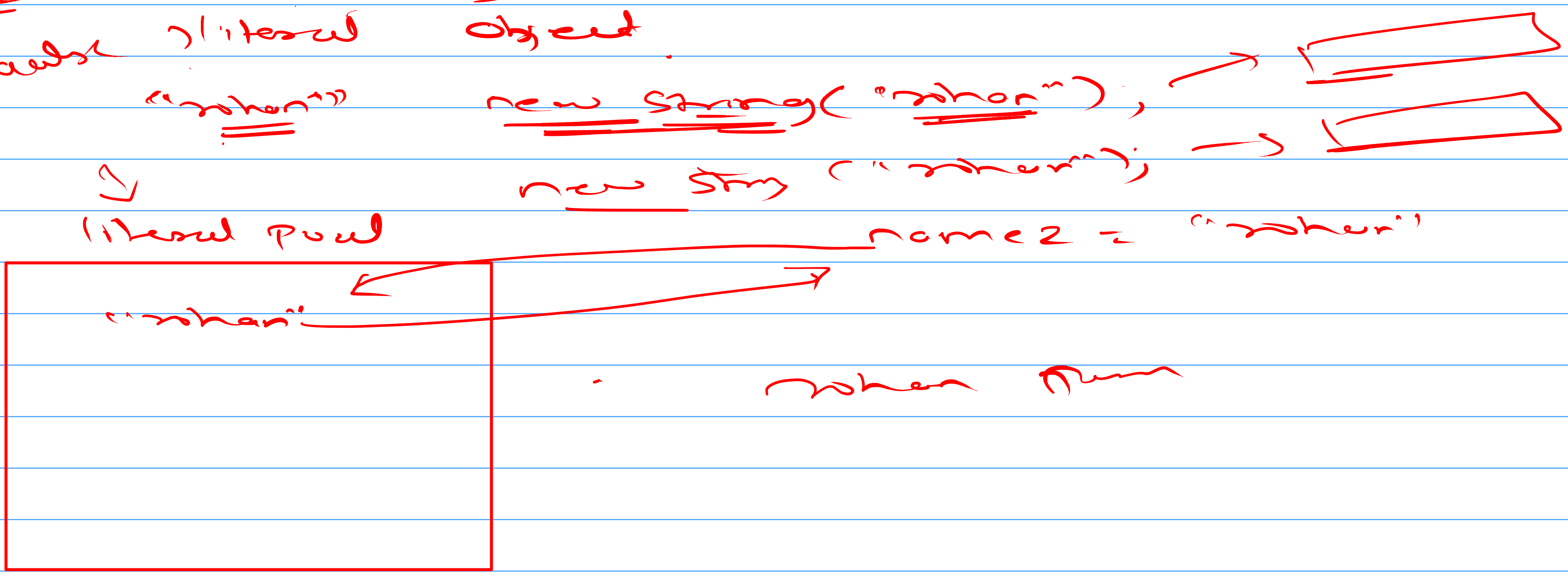
String → class → java.lang

String name → reference

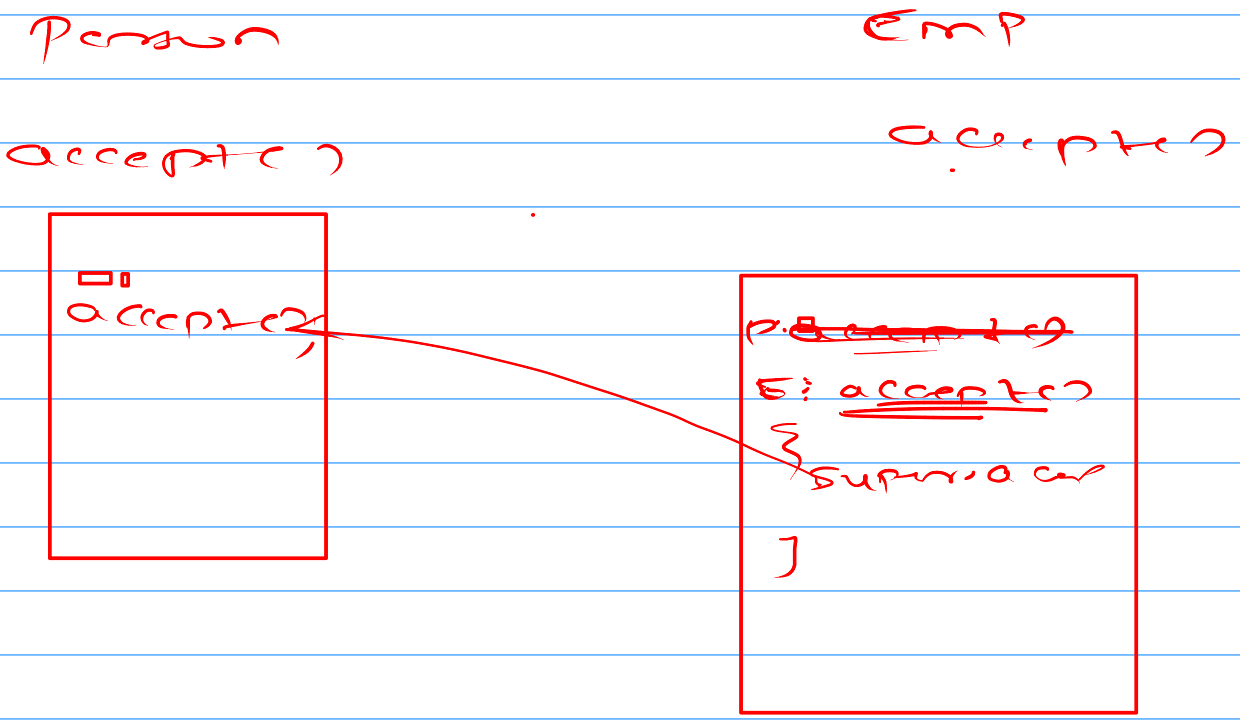
easy — literal — Object

"Rohan" — new String("Rohan"); →

new String("Rohan"); →



Rohan Ram



Person
accept()
Emp
accept()
accept();
P. ~~accept()~~
E: accept()
{
super.accept()
}