Exception Handling
try -> Check for the exception
catch -> Handle the exception
throw
throws
finally -> to close the resources

Errors, Exceptions

Checked
- Exception -> subclass -> except Runtime
- Compulsary to handle the exceptions

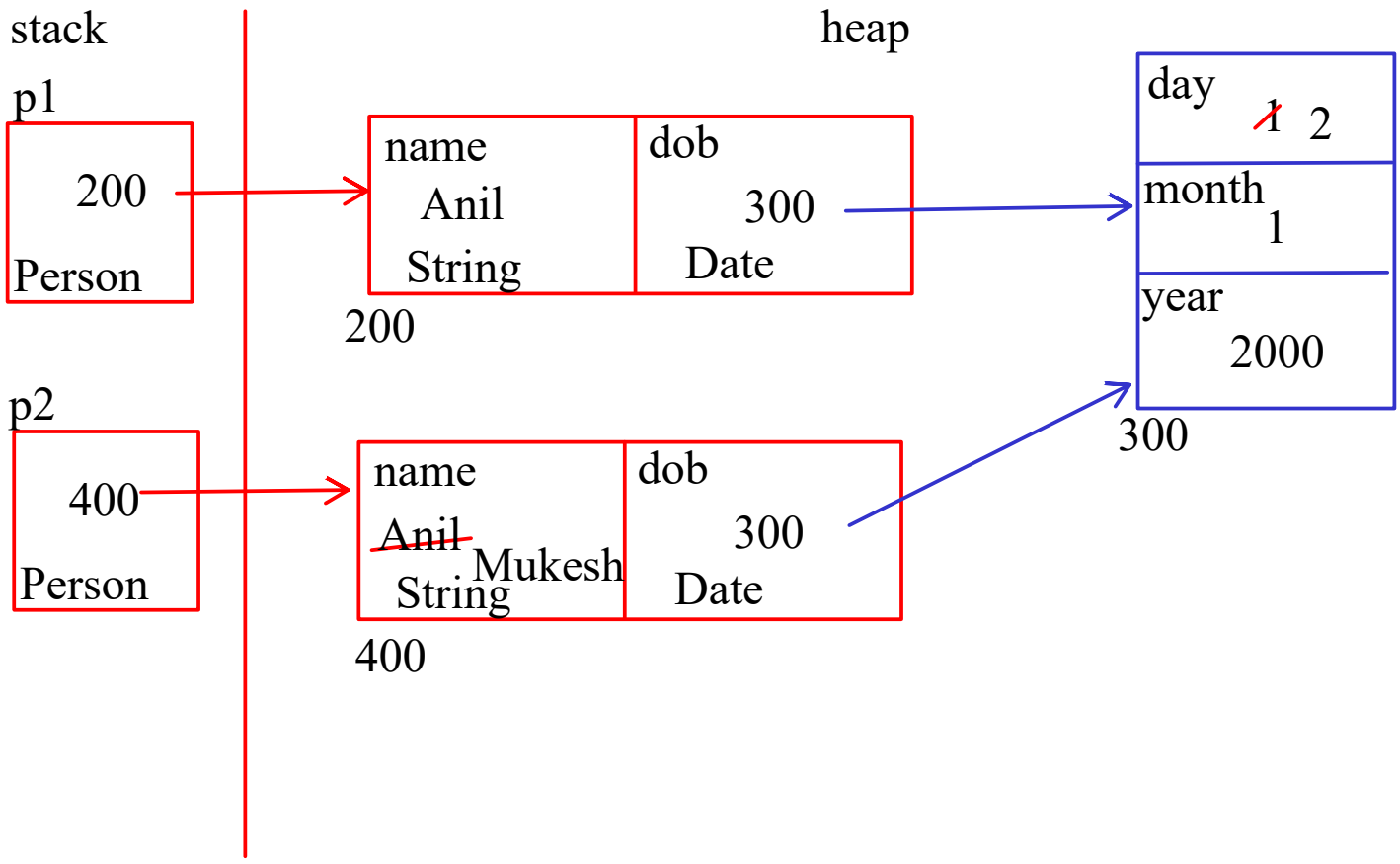
UnChecked
- Runtime -> Sub class
- It is not mandatory to handle these exceptions

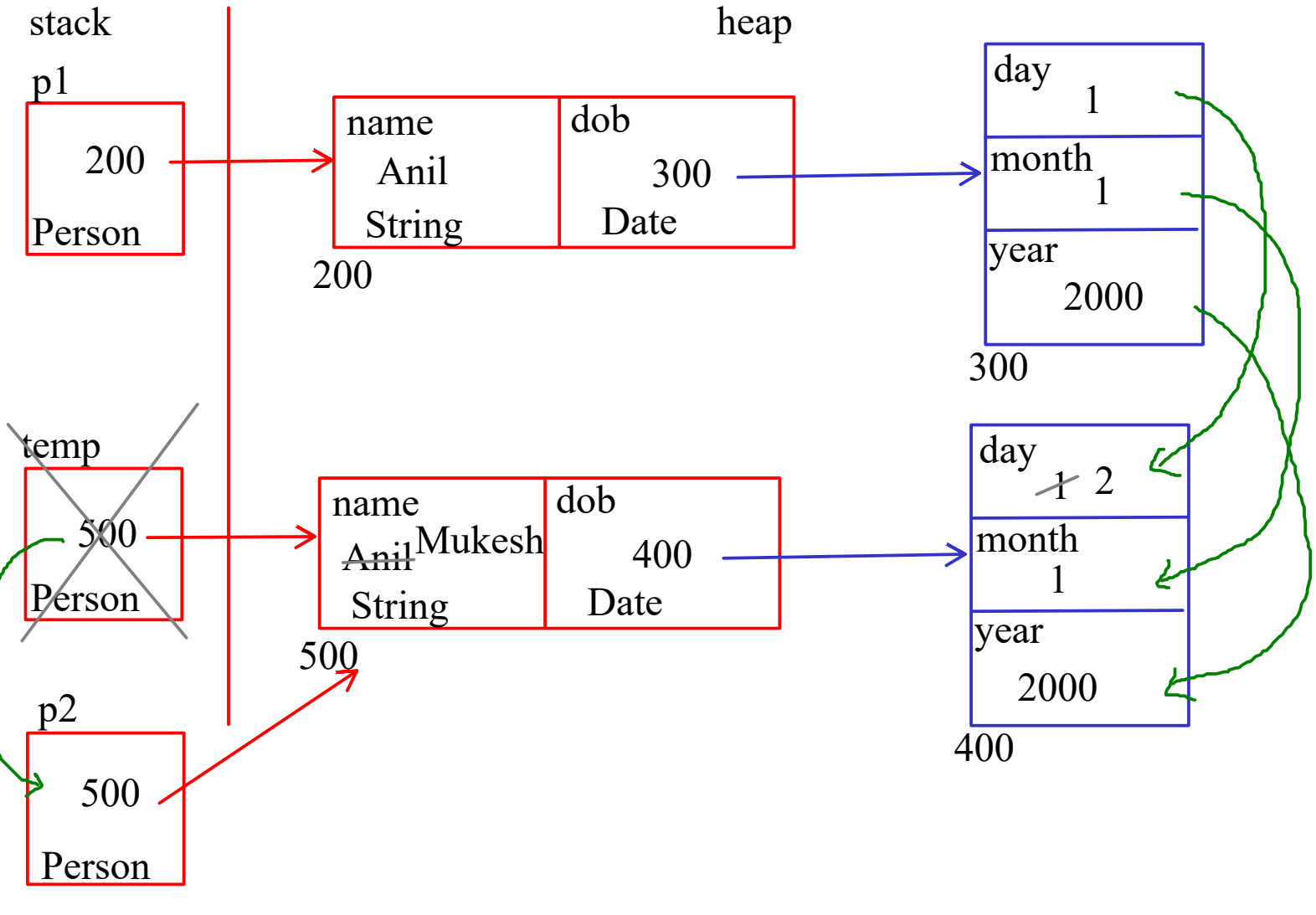
```
void doTask() throws SQLException{
// TO-DO
if(conncetion ==null)
        throw new SQLException(new ClassNotFoundException())
}
```

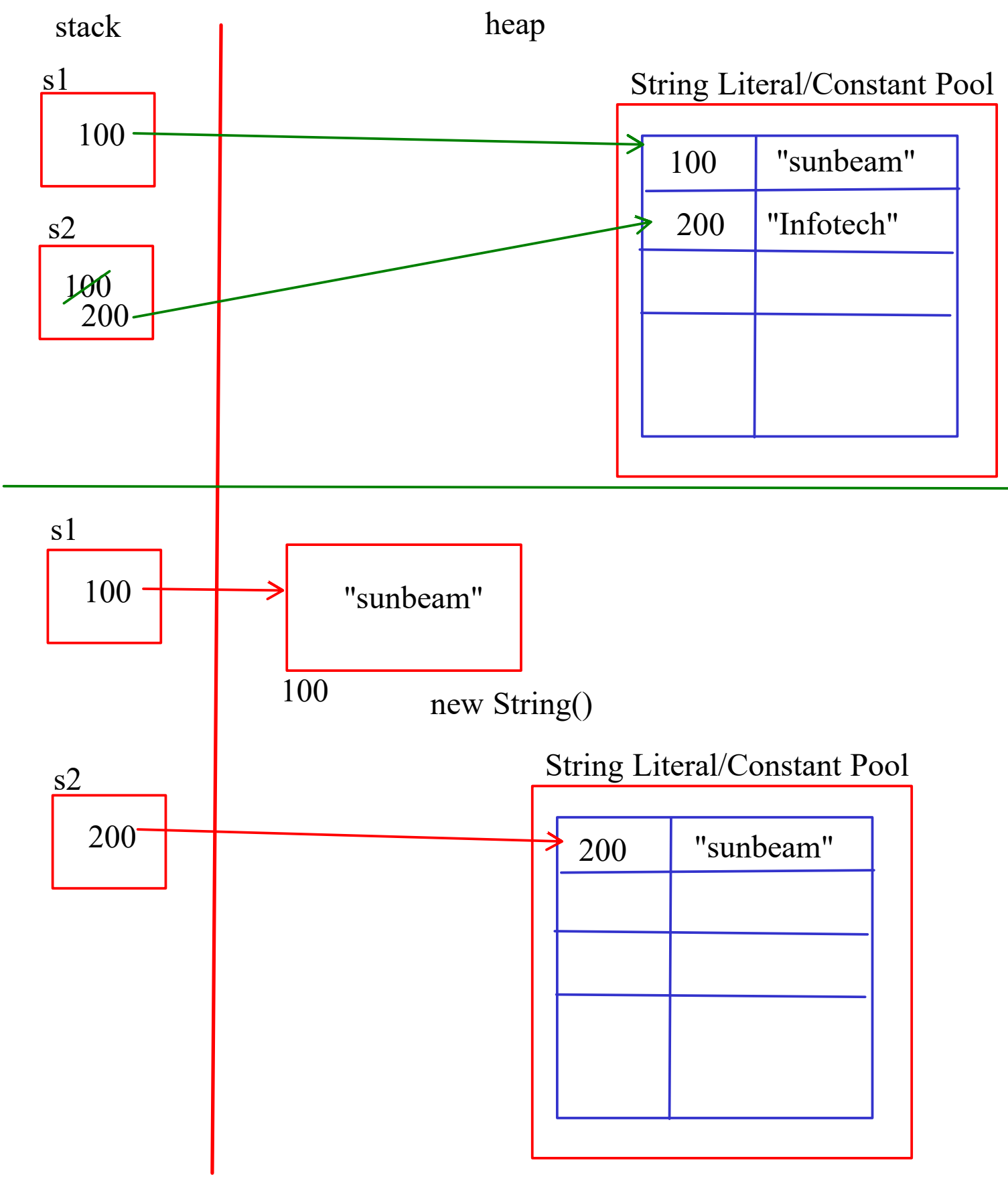
#throw
- It is used to generate new Exceptions.
- We can throw Checked or Unchecked Exception.
- To throw an checkd exception create the object of Exception class or its subclass except Runtime exception clas
- To throw unchecked exception create the object of Runtime Exception class or uts sub class
- If checked exception is thrown then we need to route the generated exception towards the caller method

throws
- throws keyword is used to route the checked exceptions towards the caller method.
- we can route the exceptions till it gets received in the main method.
- it is recommended that from main method we should not route the exceptions towrds the JVM

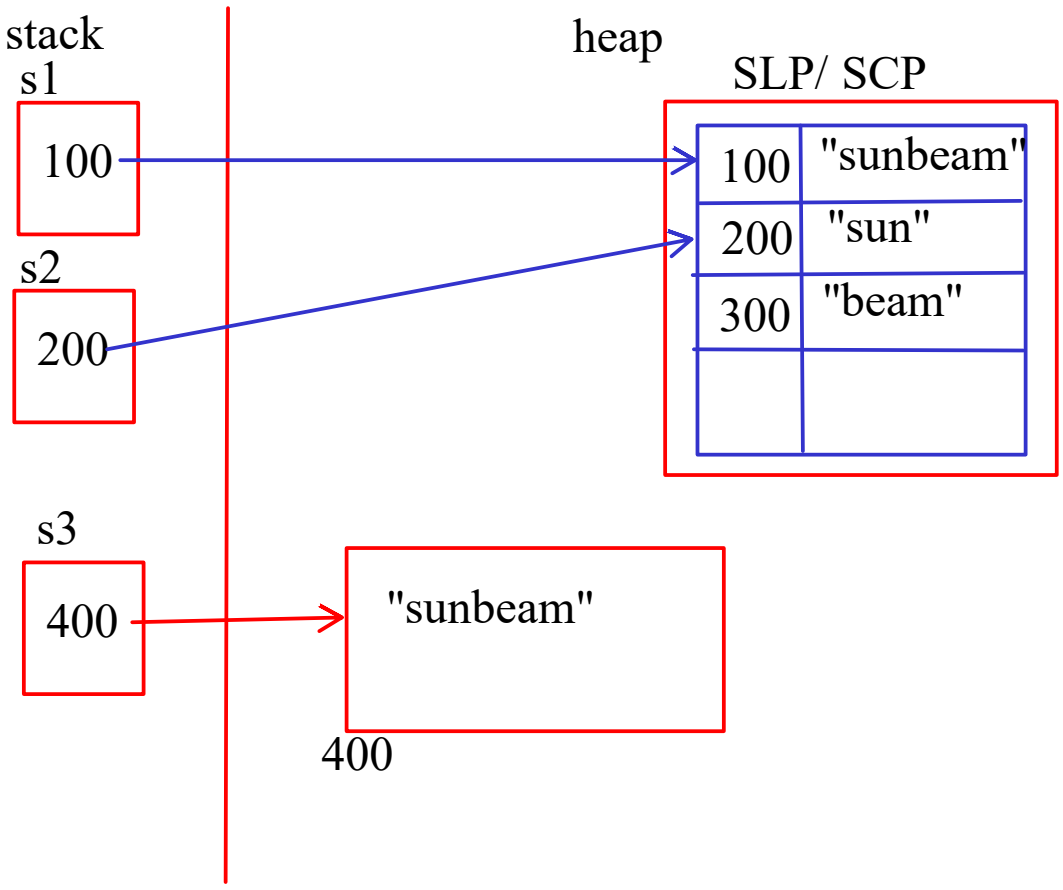
Exception Chaining
- keeping an exception object inside an another exception object is called as exception chaining

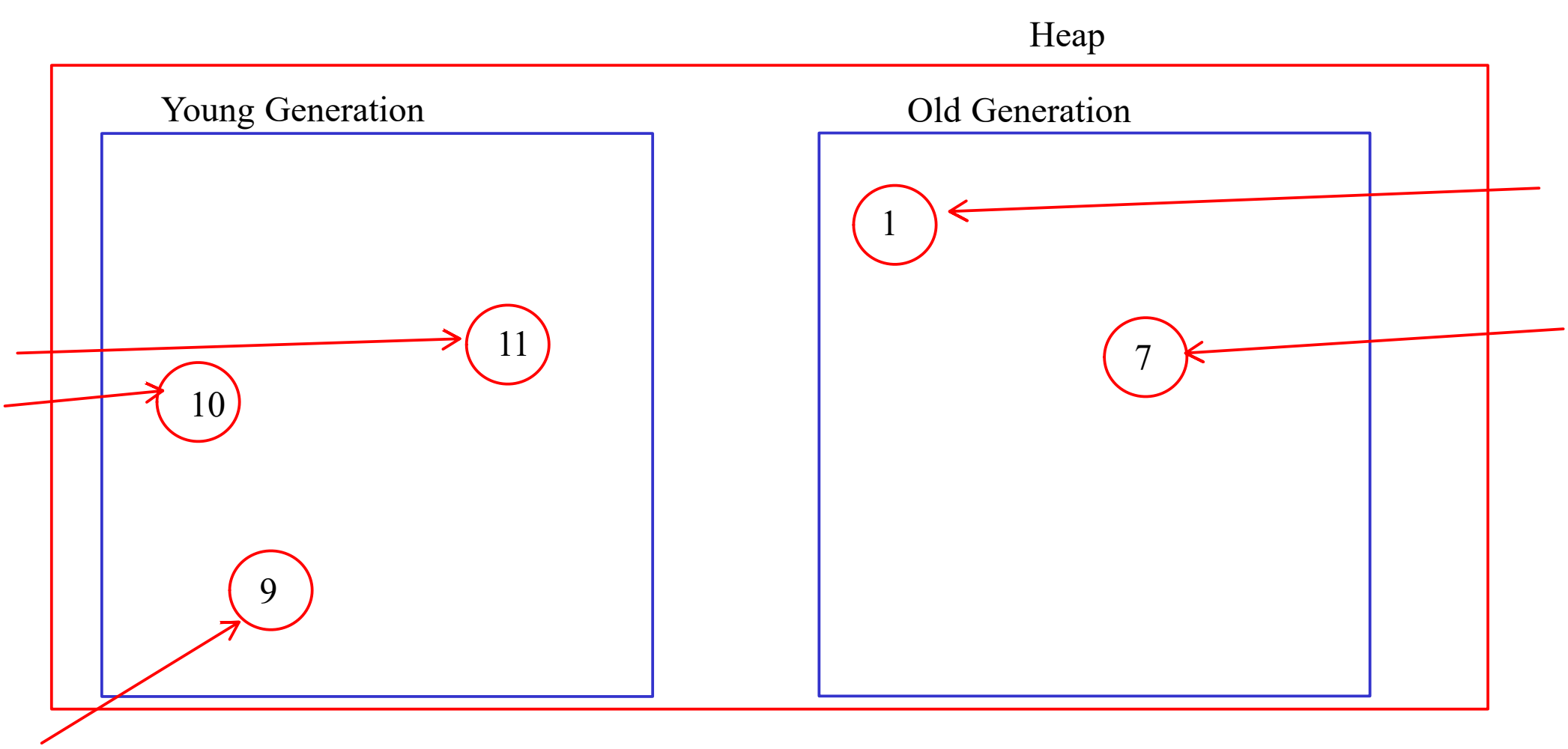






int n1 = 30;
String s2 = "sunbeam";





Date d1 = new Date(); // GC
d1 = null;

Minor GC
Major GC

```
Date d1 = new Date(); // GC
d1 = new Date();

class Program{
m1()// this{
Date d1 = new Date(); // GC
}
}

Program p = new Program();
p.m1();
```