```
int n = sc.nextInt();
Fruit[] arr = new Fruit[n];
int index = 0;

1.
if(index<n){
arr[index] = new Apple();
arr[index].accept();
index ++;
}

2.
if(index<n){
arr[index] = new Mango();
arr[index].accept();
index ++;
}

3.
if(index<n){
arr[index] = new Orange();
arr[index].accept();
index ++;
}
```

```
abstract class Fruit{
name
color
weight
isFresh = true;

Fruit(String name){
this.name = name;
}
void accept(){

}

abstract String taste();
}
```

```
class Apple{
Apple(){
super("Apple");
}
}

class Orange{
Orange(){
super("Orange");
}
}

class Mango{
Mango(){
super("Mango");
}
}
```

1,2,3,4,5,6,7,8,9,10

```
for(Fruit f:arr)
    if(f.taste().equals("sour"))
        f.setIsFresh(false);
```

Interface
- Exception
    -

Throwable
- Error
- Exception
    1. Checked
    2. UnChecked

1. try
2. catch
3. throw
4. throws
5. finally

```
try(){
}catch(){
}finally{

}
```

Custom Exceptions

String name = "sunbeam";

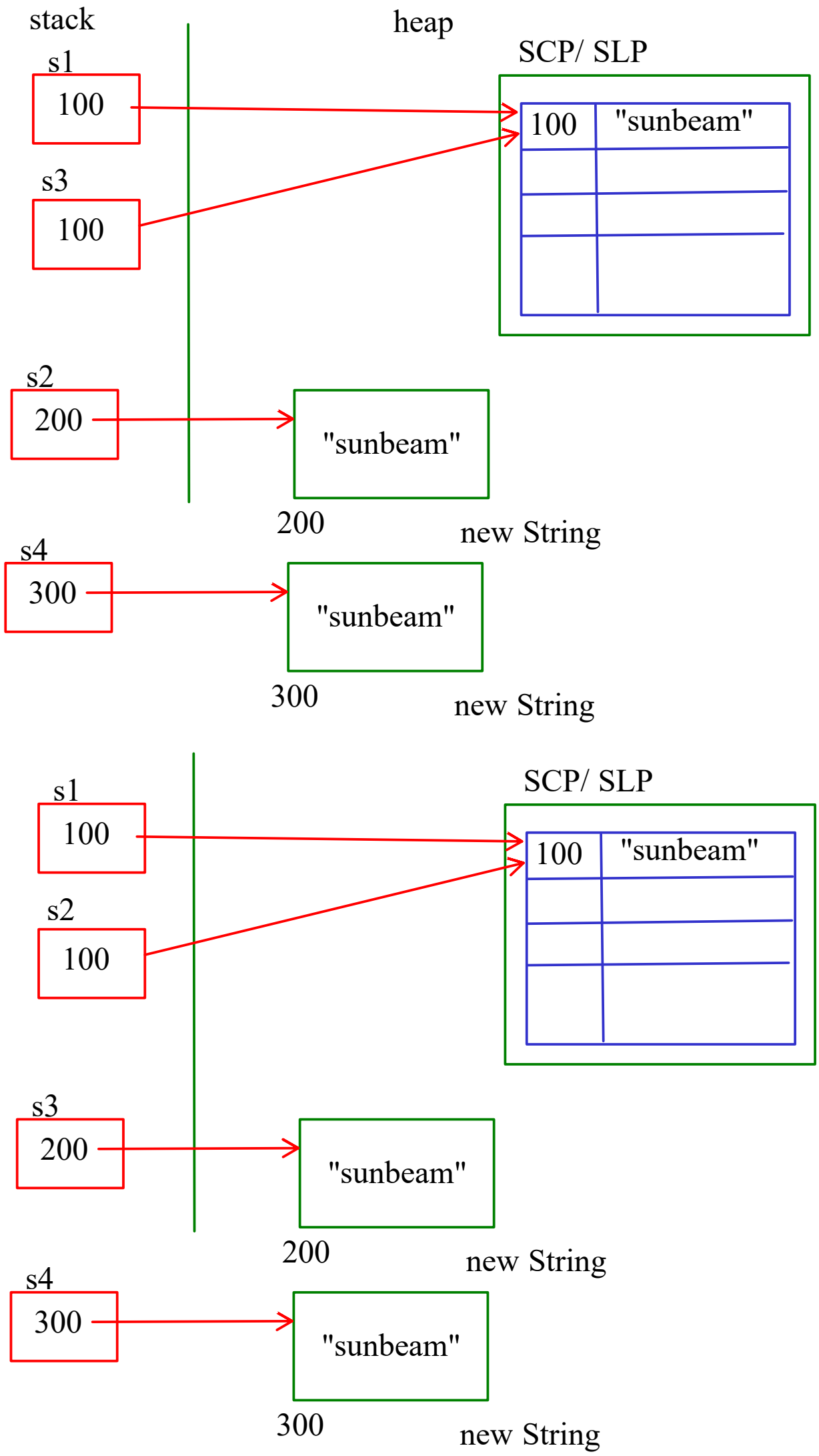
Generics
- type parameters
- Generic Iterfaces

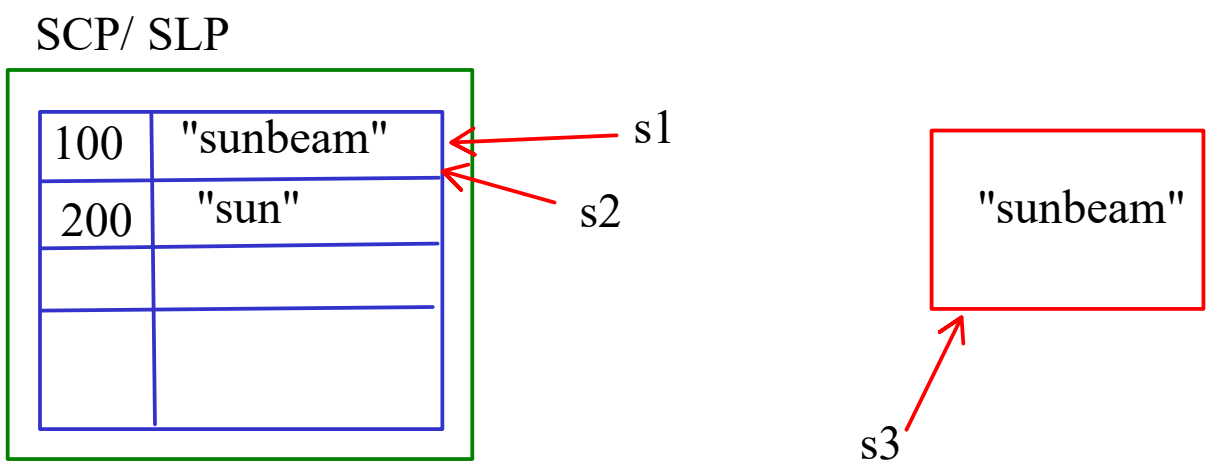
Collection Framework
Java 8 interface
Functional Interface
Lambda Expression
Stream API
Java File IO
Annotations, Reflection,
Local and Nested class,
Enum
MultiThreading

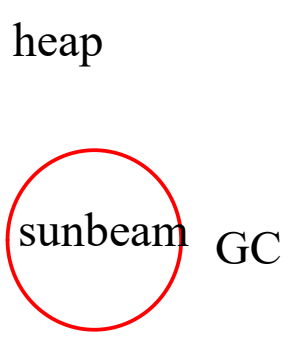
1.5 days
2 days
1 day
1.5 days
1 day
1 day

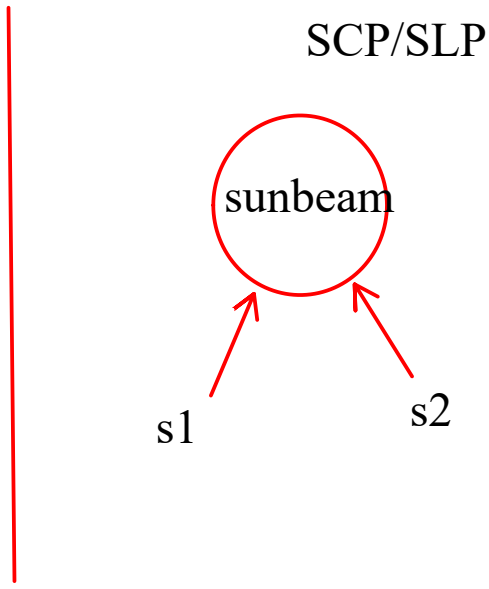
Literals/constants
1.boolean
2. character
3. integral
4. floating-point
5. string
6. null literal

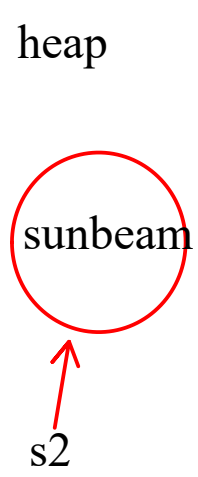
```
boolean isFresh=true
char ch = 'a';
String name = "sunbeam"; // string literal
Date d = null
```

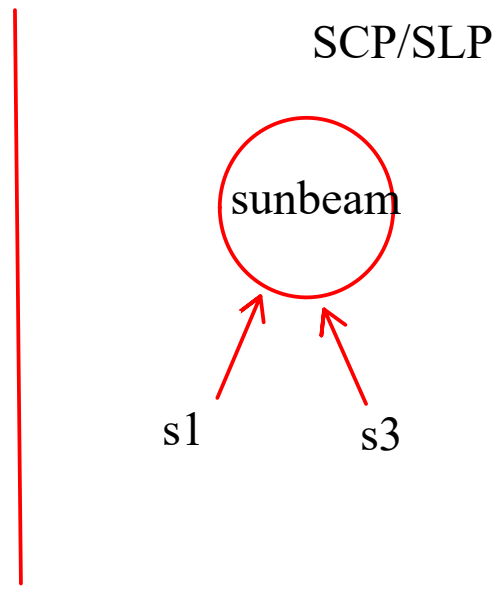

stack
heap
s1
100
SCP/ SLP
100
"sunbeam"
s3
100
s2
200
"sunbeam"
200
new String
s4
300
"sunbeam"
300
new String
s1
100
SCP/ SLP
100
"sunbeam"
s2
100
s3
200
"sunbeam"
200
new String
s4
300
"sunbeam"
300
new String

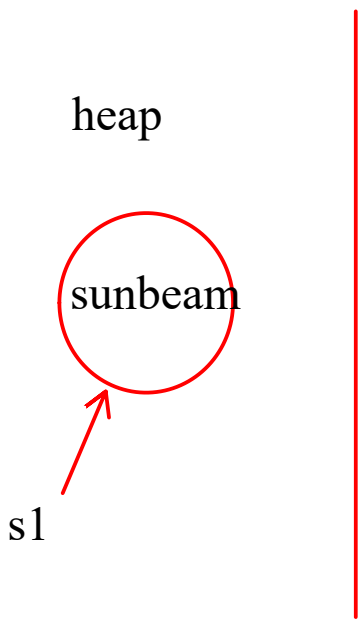

SCP/ SLP
100
"sunbeam"
200
"sun"
s1
s2
"sunbeam"
s3

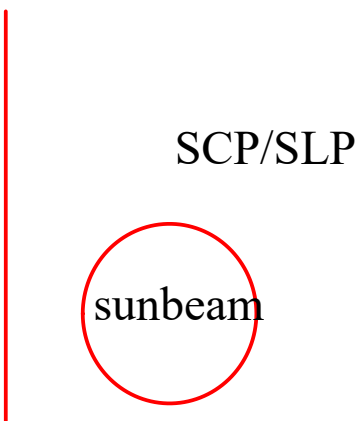


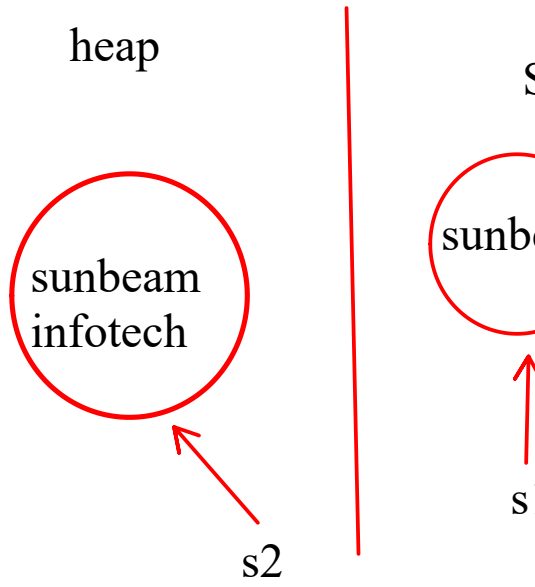


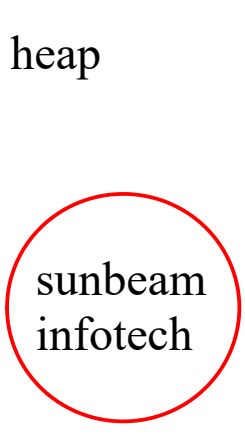


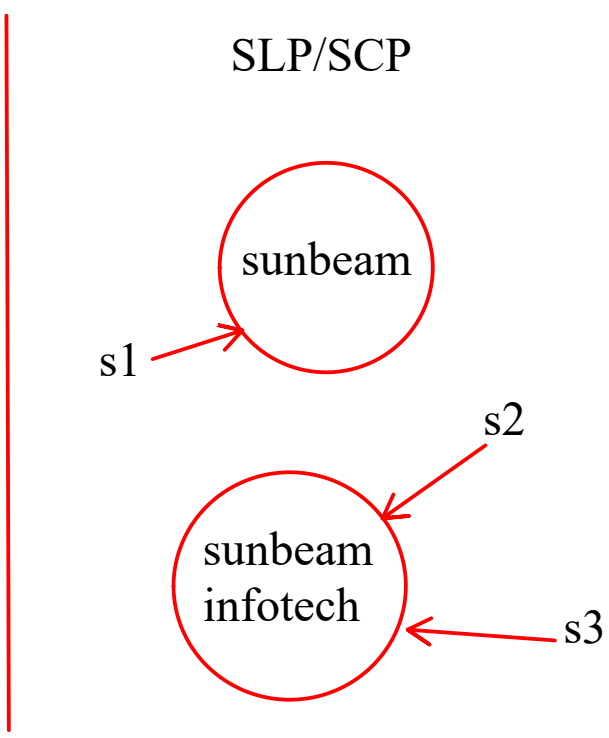


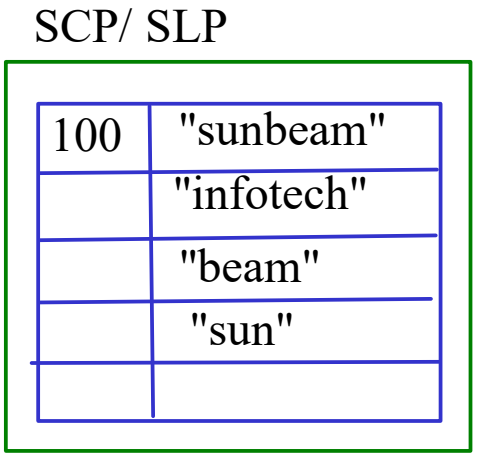













```
class Voter{
String vid;
String name;
String Address;
String mobile;
String city;

}
```

1 crore
10 lakhs -> Voters
10 lakhs -> vid objects
10 lakhs -> name objects will be < 10 lakh
10 lakhs -> address objects will be <10 lakh
10 lakhs -> mobile objects
10 lakhs -> city 1 object

```
enum ArithmeticOperations {
	EXIT, ADD, SUB, MUL
}

	final class ArithmeticOperations extends Enum{
		public static final ArithmeticOperations EXIT;
		public static final ArithmeticOperations ADD;
		public static final ArithmeticOperations SUB;
		public static final ArithmeticOperations MUL;
		private static final ArithmeticOperations [] ENUM$VALUES;

		private ArithmeticOperations(String name, int ordinal){
			super(name,ordinal);
		}
		static{
			EXIT = new ArithmeticOperations("EXIT",0);
			ADD = new ArithmeticOperations("ADD",1);
			SUB = new ArithmeticOperations("SUB",2);
			MUL = new ArithmeticOperations("MUL",3);
			ENUM$VALUES = new ArithmeticOperations[]{EXIT,ADD,SUB,MUL};
		}

		public static ArithmeticOperations[] values(){
			return ENUM$VALUES;
		}

		public static ArithmeticOperations valueof(String name){
			for(ArithmeticOperations e:ENUM$VALUES)
				if(e.name().equals(name))
					return e;
		}
	}


class Constants{
public static final String BASE_URL = "http://sunbeaminfo.com:4200/";
public static boolean LOGIN_STATUS = false;
}
```

upcasting
downcasting

class
object
reference