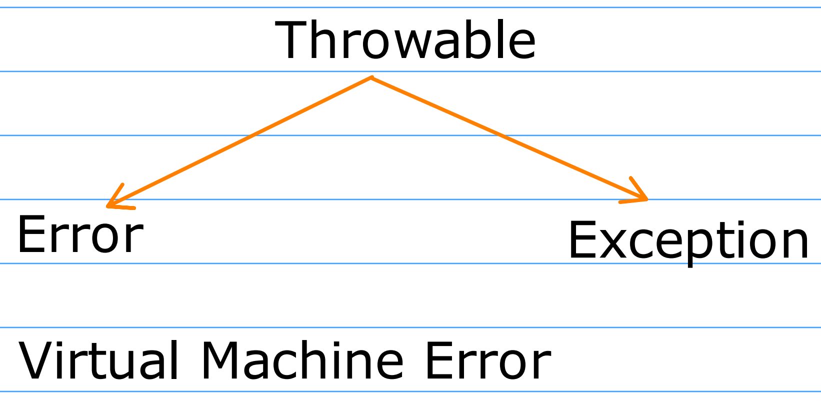
Exception Handling

- Runtime problems

1. Error -> Runtime Problems caused by Runtime Time environment
    - Java recommends not to handle the errors, let the program crash
2. Exception
    - RunTime Problems caused by Program due to wrong inputs or logical errors are considered as Exception
    - Java Recommends to handle the exception

try
catch
throw
throws
finally



```
try{                        try{                  try(){      // try-with-resource

}                           }                     }
catch(Exception e){         finally{                     x           try(){

}                           }                                        }catch(Exception e){

                                                                     }
                                                                     finally{

                                                                     }
```

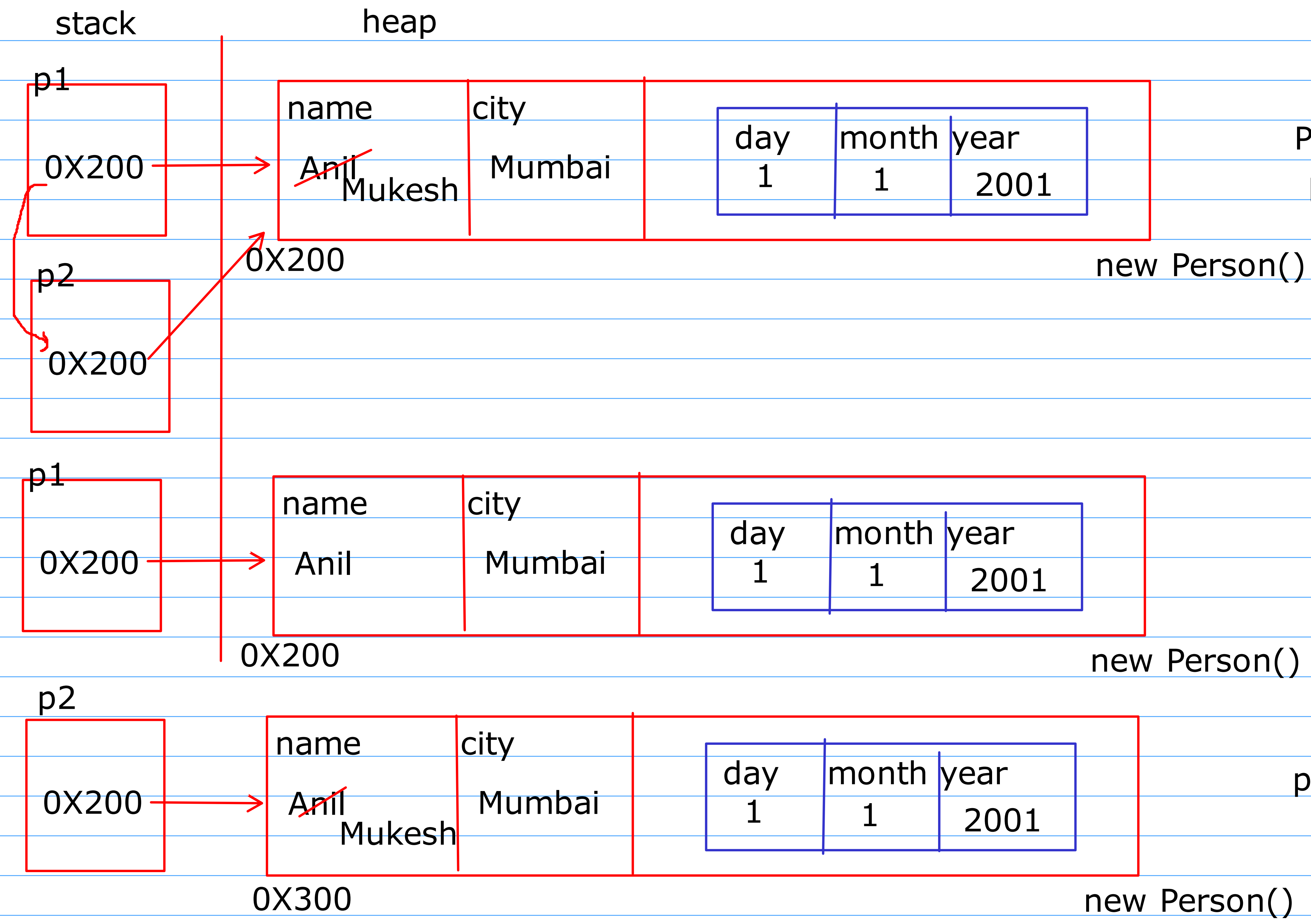
```
int n1 = 10;
Integer i1 = new Integer(n1); // Boxing
Integer i2 = 10;// Auto-Boxing
```

Checked Exception

- Exception Class itself and all its subclass except Runtime Exception class are all considered as Checked Exception
- Checked Exceptions are compulsary to handle, otherwise java will give compile time error

Unchecked Exception

- RuntimeException class and all its subclasses are all considered as UnChecked Exception
- Unchecked exceptions are optional to handle. i.e java does not give compile time error



```
Person p2 = p1;
 p2.name = "Mukesh"
```

```
p2.name = "Mukesh"
```

Shallow Copy and Deep Copy

enum -> enumeration
 - String identifiers for the constant int values

```
C              CPP
enum ....{                                      switch(choice){          switch(choice){
                                                case EXIT:               case 0:
}                          1 -> Monday                                          EXIT;
                           2 -> Tuesday  case Monday:                  case 1:
                           3
 1 -> Jan                  4             case Tuesday:                 case 2:
 2 -> Feb                  5
 3                         6
 4                         7             }                             }
 .
 .
 .                           enum Week{                              const int MON = 1;
 12 -> Dec                   MON=1,TUE,WED,.....                     const int TUE = 2;
                             }                                       const int WED = 3;
```

```
final class ArithmeticOperations extends Enum{
public static final ArithmeticOperations EXIT;
public static final ArithmeticOperations ADDITION;
public static final ArithmeticOperations SUBSTARTION;
public static final ArithmeticOperations MULTIPLICATION;
public static final ArithmeticOperations DIVISION;

static{
EXIT = new ArithmeticOperations("EXIT",0);
ADDITION = new ArithmeticOperations("ADDITION",1);
SUBSTARTION = new ArithmetiOperations();
MULTIPLICATION = new ArithmetiOperations();
DIVISION = new ArithmetiOperations();
ArithmeticOperations[] $VALUES = new ArithmeticOperations[]{
EXIT,ADDITION,SUBSTARACTION,...
}
}

private ArithmeticOperations(){
}

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