* To handle null pointer exception in better way, we use optional class
* Other we use lot of null checks in the code

Optional.of() and Oprtional.OfNullable()

* Both accept element and return optional with specified element

|  |
| --- |
| If given element is null ,then Optional.OfNullable () method return empty optional  But in Optional.Of() throws NPE when the given element is NULL |

Sample code without using optional.

Person person = new Person("sailtha", 28, "SE");  
String upperCaseName = "";  
if (person != null) {  
String name = person.getName();  
if (name != null) {  
upperCaseName = name.toUpperCase();  
}  
}  
System.out.println(upperCaseName);

Person person = new Person(null, 28, "SE");  
Optional<Person> personOptional = Optional.ofNullable(person);  
personOptional.ifPresent(p -> {  
Optional<String> name = Optional.ofNullable(p.getName());  
**String upperCaseName = name.map(String::toUpperCase).orElse("Empty Name");**  
System.out.println(upperCaseName);  
});

Like streams we can us Map and filer

List<Person> personList = Arrays.asList(  
new Person("Andrew", 28, null),  
new Person("John", 34, "QA"),  
new Person("Peter", 43, "SE"),  
new Person("Tim", 39, "TL")  
);  
List<Person> agedPersonsList = personList.stream()  
.filter(p -> p.getAge() < 30)  
.collect(Collectors.toList());  
Optional<Person> personData = Optional.ofNullable(agedPersonsList.get(0));  
personData.ifPresent(p -> {  
Optional<String> job = Optional.ofNullable(p.getJob());  
Optional<String> name = Optional.ofNullable(p.getName());  
String upperCaseJob = job.map(String::toUpperCase).orElse("Empty Job");  
String nameFilter = name.filter(n -> n.equals("Andrew")).map(String::toUpperCase).orElse("Empty name");  
System.out.println(upperCaseJob);  
System.out.println(nameFilter);  
});