```
//  BEFORE Java 8
   public static List<Customer> filterSFCustomers(List<Customer> customers) {
       List<Customer> result = new ArrayList<>();
       for (Customer customer : customers) {
           if ("SF".equals(customer.getCity())) {
               result.add(customer);
           }
       }
       return result;
   }

// parameterize the city
   public static List<Customer> filterCustomersByCity(List<Customer> customers, String city) {
       List<Customer> result = new ArrayList<>();
       for (Customer customer : customers) {
           if (customer.getCity().equals(city)) {
               result.add(customer);
           }
       }
       return result;
   }
```

Encapsulate the filter logic in a separate class:

```
public class CustomerByCity {
        public boolean test(Customer cust, String searchStr) {
        return cust.getCity().equals(searchStr);
    }
```

Pass an Object of Type CustomerByCity to the search method:

```
List<Customer> filteredCusts = filterCustomers(custs, new CustomerByCity(), "LA");

 public static List<Customer> filterCustomers(List<Customer> customers, CustomerByCity cbc,
String searchStr) {
        List<Customer> result = new ArrayList<>();
        for (Customer customer : customers) {
            if (cbc.test(customer, searchStr)){
                result.add(customer);
            }
        }
        return result;
    }
```

Why create a physically separate class? Use an anonymous class instead:

```
filteredCusts = filterCustomers(custs,
        new CustomerPredicate(){
          public boolean test(Customer cust, String searchStr){
               return cust.getCity().equals(searchStr);
          }
      }, "LA");
```

Can we limit the amount of code? Use a Functional Interface instead. It must have same return type and parameter list as the current method:

```
public interface CustomerPredicate {
       boolean test(Customer cust, String searchStr);
}
```

```
filteredCusts = filterCustomers(custs,
(Customer cust, String searchStr) -> cust.getCity().equals(searchStr), "LA");
```

```
// use CustomerPredicate
        public static List<Customer> filterCustomers(List<Customer> customers,
CustomerPredicate custPred, String searchStr) {
        List<Customer> result = new ArrayList<>();
        for (Customer customer : customers) {
            if (custPred.test(customer, searchStr)){
                result.add(customer);}
```