```java
package ch14;

import java.util.ArrayList;
import java.util.Optional;

public class OptionalEx0 {


	//Option이해하기 위해  Optional을 사용하지 않는 코드와 비교하자
	//고객을 검색시 없는 경우 null을 반환하게 된다.  이때 반환된 객체를 사용하는 코드가 있다면
	//Nullpoint Exception을 만나게 된다
	//반환객체를 Optional객체로 감싸서 반환하게 된다.
	//null을 직접다루지 않겠다는 것이 Optional 의 존재이유

	public static void main(String[] args) {
		Optional<User> user = searchUserOptional("gy");
		//user객체가 비어있다면 빈객체를 만들어서 사용하라는 코드임

		//검색결과가 없을 때의 코드를 작성할 수 있다
		System.out.println(user.orElse(new User()).getId());
		user.ifPresentOrElse( value -> System.out.println( "결과" +  value),
				() -> System.out.println("검색결과없음"));


		//Optional을 사용하지 않는 코드
		//User user2 = searchUser("gy");
		//System.out.println(user2.getId()); // NullPointException
	}

	private static Optional<User> searchUserOptional(String id) {
		ArrayList<User> users = new ArrayList<>();
		users.add(new User("hong", "t11"));
		users.add(new User("kim", "t11"));
		users.add(new User("lee", "t11"));
		return users.stream().filter(user -> user.getId().equals(id)).findFirst();
	}

	private static User searchUser(String id) {
		User user = null;
		ArrayList<User> users = new ArrayList<>();
		users.add(new User("hong", "t11"));
		users.add(new User("kim", "t11"));
		users.add(new User("lee", "t11"));
		for (User u : users) {
			if (u.getId().equals(id)) {
				user = u;
			}
		}
		return user; // 못찾으면 null 반환
	}
}
```