OneToMany-Bi-Directional Mapping
==========

1. Person.java

```
@Entity
@Setter
@Getter
@AllArgsConstructor
@Table(name = "OTM_PERSON")
@RequiredArgsConstructor
public class Person {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private Integer pid;

	@NonNull
	private String pname;

	@NonNull
	private String paddress;

	@OneToMany(targetEntity = PhoneNumber.class, cascade = CascadeType.ALL)
	@JoinColumn(name = "PERSON_ID", referencedColumnName = "pid")
	private Set<PhoneNumber> contactDetails;

	Person() {
		System.out.println("Person 0 param constructor :: " + this.getClass());
	}

	@Override
	public String toString() {
		return "Person [pid=" + pid + ", pname=" + pname + ", paddress=" +
paddress + "]";
	}

}
```

2. PhoneNumber.java

```
@Entity
@Setter
@Getter
@AllArgsConstructor
@Table(name = "OTM_PHONE_NUMBER")
@RequiredArgsConstructor
public class PhoneNumber {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long regNo;

	@NonNull
	private Long phoneNo;

	@NonNull
	private String provider;
```

```
	@NonNull
	private String type;

	@ManyToOne(targetEntity = Person.class, cascade = CascadeType.ALL)
	@JoinColumn(name = "PERSON_ID", referencedColumnName = "pid")
	private Person person;

	@Override
	public String toString() {
		return "PhoneNumber [regNo=" + regNo + ", phoneNo=" + phoneNo + ",
provider=" + provider + ", type=" + type
				+ "]";
	}

	PhoneNumber() {
		System.out.println("PhoneNumber:: Zero param constructor is called ::"
+ this.getClass());
	}

}
```

Note:: In Bi-Directional Association instead of specifying foreign key column using @JoinColumn in both parent and hild class, we can specify only
at one side with the support of "mappedBy" param.
In OnetoMany Association we need to specify mappedby at Oneside(Parent class).
In ManytoMany Association we can specify mappedBy at any side(Parent class).

```
1. @JoinColumn(name="PERSON_ID",referenceColumnName = "pid")
		PERSON_ID
		  ;;;;

2.
@OnetoMany(targetEntity=PhoneNumber.class,cascade=CascadeType.all,mappedBy="person"
)
		person_pid
		  ;;;;;
```

Note:: In Association Mapping the cascading type are related to non-select operation, which indicates any non-select operation done on main object
will be cascaded/propogated to the associated child objects

In Association Mapping,the fetch types are related to select operation which indicates the associated child objects should be loaded along with
the Parent objects or not.
		FetchType.EAGER => child objects will loaded along with parent object
		FetchType.LAZY => Parent objects will loaded normally,but associated child objects will be loaded lazily on demand basis.

=> Default fetch type for OnetoMany,OnetoOne,ManytoMany Association :: LAZY
=> Defalut fetch type for Many to One Association :: EAGER

Delete Operation in Association Mapping
==========
Delete Parent object, so Deletion of Child object also should happen.
Delete child object, but parent object should not be deleted.

Joins
=====
 => Joins are given to get data from two db table of association having some implicit condition
 => We can also add new condition on top of implicit condition
 => In SQL to work with joins the two db table need not be in relationship, whereas in HQL/JPQL the two dbtalbles needs to be in association to apply joins
 => To work with HQL/JPQL we need to keep dbtables and their entity class in relationship using association mapping concepts.

Syntax::
  select <parent class propertiye>,<child class properties> from <parent class><alias name>
                                        <join type> <parent class HAS-A property> <aliasname> <additional condition>

IPersonRepo.java
================
```
public interface IPersonRepo extends JpaRepository<Person, Integer>{
	//@Query("select p.pid,p.pname,p.paddress,ph.regNo,ph.phoneNo,ph.provider,ph.type  from Person p inner join  p.contactDetails ph")
	//@Query("select p.pid,p.pname,p.paddress,ph.regNo,ph.phoneNo,ph.provider,ph.type  from Person p right join  p.contactDetails ph")
	//@Query("select p.pid,p.pname,p.paddress,ph.regNo,ph.phoneNo,ph.provider,ph.type  from Person p left join  p.contactDetails ph")
	@Query("select p.pid,p.pname,p.paddress,ph.regNo,ph.phoneNo,ph.provider,ph.type  from Person p full join  p.contactDetails ph")//Error
	public List<Object[]> fetchDataUsingJoinsByParent();

}
```

AssociationRunner.java
======================
```
@Component
public class AssociationRunner implements CommandLineRunner {

	@Autowired
	private IPersonMgmtService service;

	@Override
	public void run(String... args) throws Exception {
		service.fetchDataByJoinsUsingParent().forEach(row->{
			for (Object obj : row) {
				System.out.print(obj+" ");
			}
			System.out.println();
		});
	}
}
```