PersonController.java

```
@RestController
@RequestMapping("/api/v1/people")
@AllArgsConstructor(onConstructor = @__(@Autowired))
public class PersonController {
```

message: "__ is not an annotation type",

message: "The value for annotation attribute AllArgsConstructor.onConstructor must be some @lombok.AllArgsConstructor.AnyAnnotation annotation”,

message: "__ cannot be resolved to a type",

PhoneType.java

```
@Getter
@AllArgsConstructor
public enum PhoneType {

    HOME("Home"),
    MOBILE("Mobile"),
    COMMERCIAL("Commercial");

    private PhoneType() {
    }

    private final String description;
}
```

message: "The constructor PhoneType(String) is undefined",

PersonService.java

```
@Service
@AllArgsConstructor(onConstructor = @__(@Autowired))
public class PersonService {
```

message: "The value for annotation attribute AllArgsConstructor.onConstructor must be some @lombok.AllArgsConstructor.AnyAnnotation annotation”,

message: "_cannot be resolved to a type,

PersonService.java

```
 private PersonRepository personRepository;

  private final PersonMapper personMapper = PersonMapper.INSTANCE;

  public MessageResponseDTO createPerson(PersonDTO personDTO) {
     Person personToSave = personMapper.toModel(personDTO);

     Person savedPerson = personRepository.save(personToSave);
     return createMessageResponse(savedPerson.getId(), "Created person with ID ");
  }
```

message: "The method getId() is undefined for the type Person”,

```
public MessageResponseDTO updateById(Long id, PersonDTO personDTO) throws PersonNot
FoundException {
     verifyIfExists(id);

     Person personToUpdate = personMapper.toModel(personDTO);

     Person updatedPerson = personRepository.save(personToUpdate);
     return createMessageResponse(updatedPerson.getId(), "Updated person with ID ");
  }
```

message: "The method getId() is undefined for the type Person”,

```
private MessageResponseDTO createMessageResponse(Long id, String message) {
     return MessageResponseDTO
          .builder()
          .message(message + id)
          .build();
  }
}
```

message: "The method builder() is undefined for the type MessageResponseDTO”,

PersonServiceTest.java

```
@Test
  void testGivenPersonDTOThenReturnSavedMessage() {
    PersonDTO personDTO = createFakeDTO();
    Person expectedSavedPerson = createFakeEntity();

    when(personRepository.save(any(Person.class))).thenReturn(expectedSavedPerson);

    MessageResponseDTO expectedSuccessMessage = createExpectedMessageResponse(expec
tedSavedPerson.getId());
    MessageResponseDTO succesMessage = personService.createPerson(personDTO);

    assertEquals(expectedSuccessMessage, succesMessage);
  }
```

message: "The method getId() is undefined for the type Person”

```
private MessageResponseDTO createExpectedMessageResponse(Long id) {
    return MessageResponseDTO
        .builder()
        .message("Created person with ID " + id)
        .build();
  }
}
```

message: "The method builder() is undefined for the type MessageResponseDTO”,

PersonUtils.java

```
public static PersonDTO createFakeDTO() {
    return PersonDTO.builder()
        .firstName(FIRST_NAME)
        .lastName(LAST_NAME)
        .cpf(CPF_NUMBER)
        .birthDate("04-04-2010")
        .phones(Collections.singletonList(PhoneUtils.createFakeDTO()))
        .build();
  }
```

message: "The method builder() is undefined for the type PersonDTO”

PersonUtils.java

```
public static Person createFakeEntity() {
    return Person.builder()
            .id(PERSON_ID)
            .firstName(FIRST_NAME)
            .lastName(LAST_NAME)
            .cpf(CPF_NUMBER)
            .birthDate(BIRTH_DATE)
            .phones(Collections.singletonList(PhoneUtils.createFakeEntity()))
            .build();
    }
}
```

message: "The method builder() is undefined for the type Person”,

PhoneUtils.java

```
public class PhoneUtils {

    private static final String PHONE_NUMBER = "1199999-9999";
    private static final PhoneType PHONE_TYPE = PhoneType.MOBILE;
    private static final long PHONE_ID = 1L;

    public static PhoneDTO createFakeDTO() {
        return PhoneDTO.builder()
                .number(PHONE_NUMBER)
                .type(PHONE_TYPE)
                .build();
    }

    public static Phone createFakeEntity() {
        return Phone.builder()
                .id(PHONE_ID)
                .number(PHONE_NUMBER)
                .type(PHONE_TYPE)
                .build();
    }
}
```

message: "The method number(String) is undefined for the type Object”,

message: "The method builder() is undefined for the type Phone”.