[Template for weekly studio submission]

Studio 4

Name and Student Id: Ke Zhang, 29218969

Self-Evaluation {To be highlighted by Student only}:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Need Help | Work in Progress | Pass | Credit | Distinction | High Distinction |

Tutor Evaluation {To be highlighted by Tutor only}:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Incomplete | Needs improvement | Pass | Credit | Distinction | High Distinction |

Task 1:

Modified PropertyRepository.java file highlighting changes with comments

1. **package** fit5042.tutex.repository;
3. **import** java.util.List;
4. **import** java.util.Set;
5. **import** javax.ejb.Remote;
7. **import** fit5042.tutex.repository.entities.ContactPerson;
8. **import** fit5042.tutex.repository.entities.Property;
10. /\*\*
11. \* @autor Eddie Leung
12. \*/
13. @Remote
14. **public** **interface** PropertyRepository {
16. /\*\*
17. \* Add the property being passed as parameter into the repository
18. \*
19. \* @param property - the property to add
20. \*/
21. **public** **void** addProperty(Property property) **throws** Exception;
23. /\*\*
24. \* Search for a property by its property ID
25. \*
26. \* @param id - the propertyId of the property to search for
27. \* @return the property found
28. \*/
29. **public** Property searchPropertyById(**int** id) **throws** Exception;
31. /\*\*
32. \* Return all the properties in the repository
33. \*
34. \* @return all the properties in the repository
35. \*/
36. **public** List<Property> getAllProperties() **throws** Exception;
38. /\*\*
39. \* Return all the contact people in the repository
40. \*
41. \* @return all the contact people in the repository
42. \*/
43. **public** List<ContactPerson> getAllContactPeople() **throws** Exception;
45. /\*\*
46. \* Remove the property, whose property ID matches the one being passed as
47. \* parameter, from the repository
48. \*
49. \* @param propertyId - the ID of the property to remove
50. \*/
51. **public** **void** removeProperty(**int** propertyId) **throws** Exception;
53. /\*\*
54. \* Update a property in the repository
55. \*
56. \* @param property - the updated information regarding a property
57. \*/
58. **public** **void** editProperty(Property property) **throws** Exception;
60. /\*\*
61. \* Search for properties whose price is less than or equal to a budget
62. \*
63. \* @param budget - the budget that the price of the returned properties must
64. \* be lower than or equal to
65. \* @return the properties found
66. \*/
67. **public** List<Property> searchPropertyByBudget(**double** budget) **throws** Exception;
69. /\*\*
70. \* Search for properties by their contact person
71. \*
72. \* @param contactPerson - the contact person that is responsible for the
73. \* properties
74. \* @return the properties found
75. \*/
76. Set<Property> searchPropertyByContactPerson(ContactPerson contactPerson) **throws** Exception;
77. }

Modified JPAPropertyRepositoryImpl.java file highlighting changes with comments

1. **package** fit5042.tutex.repository;
3. **import** fit5042.tutex.repository.entities.ContactPerson;
4. **import** fit5042.tutex.repository.entities.Property;
5. **import** java.util.List;
6. **import** java.util.Set;
7. **import** javax.ejb.Stateless;
8. **import** javax.persistence.EntityManager;
9. **import** javax.persistence.PersistenceContext;
10. **import** javax.persistence.TypedQuery;
11. **import** javax.persistence.criteria.CriteriaBuilder;
12. **import** javax.persistence.criteria.CriteriaQuery;
13. **import** javax.persistence.criteria.Predicate;
14. **import** javax.persistence.criteria.Root;
16. /\*\*
17. \*
18. \* @author Eddie Leung
19. \*/
20. @Stateless
21. **public** **class** JPAPropertyRepositoryImpl **implements** PropertyRepository {
23. //insert code (annotation) here to use container managed entity manager to complete these methods
24. @PersistenceContext(unitName = "W4ExeSolution-ejbPU")
25. **private** EntityManager entityManager;
27. @Override
28. **public** **void** addProperty(Property property) **throws** Exception {
29. List<Property> properties = entityManager.createNamedQuery(Property.GET\_ALL\_QUERY\_NAME).getResultList();
30. property.setPropertyId(properties.get(0).getPropertyId() + 1);
31. entityManager.persist(property);
32. }
34. @Override
35. **public** Property searchPropertyById(**int** id) **throws** Exception {
36. Property property = entityManager.find(Property.**class**, id);
37. property.getTags();
38. **return** property;
39. }
41. @Override
42. **public** List<Property> getAllProperties() **throws** Exception {
43. **return** entityManager.createNamedQuery(Property.GET\_ALL\_QUERY\_NAME).getResultList();
44. }
46. @Override
47. **public** Set<Property> searchPropertyByContactPerson(ContactPerson contactPerson) **throws** Exception {
48. contactPerson = entityManager.find(ContactPerson.**class**, contactPerson.getConactPersonId());
49. contactPerson.getProperties().size();
50. entityManager.refresh(contactPerson);
52. **return** contactPerson.getProperties();
53. }
55. @Override
56. **public** List<ContactPerson> getAllContactPeople() **throws** Exception {
57. **return** entityManager.createNamedQuery(ContactPerson.GET\_ALL\_QUERY\_NAME).getResultList();
58. }
60. @Override
61. **public** **void** removeProperty(**int** propertyId) **throws** Exception {
62. //complete this method
63. entityManager.remove(entityManager.find(Property.**class**, propertyId));
64. }
66. @Override
67. **public** **void** editProperty(Property property) **throws** Exception {
68. entityManager.merge(property);
69. }
71. @Override
72. **public** List<Property> searchPropertyByBudget(**double** budget) **throws** Exception {
73. //complete this method using Criteria API
74. CriteriaBuilder builder = entityManager.getCriteriaBuilder();
75. CriteriaQuery cQuery = builder.createQuery(List.**class**);
76. Root<Property> p = cQuery.from(Property.**class**);
77. cQuery.select(p);
78. Predicate predicate = builder.lessThanOrEqualTo(p.get("price").as(**double**.**class**), budget);
79. cQuery.where(predicate);
80. TypedQuery tQuery = entityManager.createQuery(cQuery);
81. **return** tQuery.getResultList();
82. }
83. }

Modified ***Address.java*** file highlighting changes with comments

1. **package** fit5042.tutex.repository.entities;
3. **import** java.io.Serializable;
4. **import** javax.persistence.Access;
5. **import** javax.persistence.AccessType;
6. **import** javax.persistence.Column;
7. **import** javax.persistence.Embeddable;
9. /\*\*
10. \*
11. \* @author Eddie Leung
12. \*/
13. @Embeddable
14. @Access(AccessType.PROPERTY)
15. **public** **class** Address **implements** Serializable {
17. **private** String streetNumber;
18. **private** String streetAddress;
19. **private** String suburb;
20. **private** String postcode;
21. **private** String state;
23. **public** Address() {
24. }
26. **public** Address(String streetNumber, String streetAddress, String suburb, String postcode, String state) {
27. **this**.streetNumber = streetNumber;
28. **this**.streetAddress = streetAddress;
29. **this**.suburb = suburb;
30. **this**.postcode = postcode;
31. **this**.state = state;
32. }
34. @Column(name = "street\_number")
35. **public** String getStreetNumber() {
36. **return** streetNumber;
37. }
39. **public** **void** setStreetNumber(String streetNumber) {
40. **this**.streetNumber = streetNumber;
41. }
43. //complete by adding annotation here for column name = "street\_address"
44. @Column(name = "street\_address")
45. **public** String getStreetAddress() {
46. **return** streetAddress;
47. }
49. **public** **void** setStreetAddress(String streetAddress) {
50. **this**.streetAddress = streetAddress;
51. }
52. @Column(name = "suburb")
53. **public** String getSuburb() {
54. **return** suburb;
55. }
57. **public** **void** setSuburb(String suburb) {
58. **this**.suburb = suburb;
59. }
61. //complete by adding annotation here for column name = "postcode"
62. @Column(name = "postcode")
63. **public** String getPostcode() {
64. **return** postcode;
65. }
67. **public** **void** setPostcode(String postcode) {
68. **this**.postcode = postcode;
69. }
70. @Column(name = "state")
71. **public** String getState() {
72. **return** state;
73. }
75. **public** **void** setState(String state) {
76. **this**.state = state;
77. }
79. @Override
80. **public** String toString() {
81. **return** streetNumber + " " + streetAddress + ", " + suburb + ", " + state + " " + postcode;
82. }
83. }

Modified ***ContactPerson.java*** file highlighting changes with comments

1. **package** fit5042.tutex.repository.entities;
3. **import** java.io.Serializable;
4. **import** java.util.HashSet;
5. **import** java.util.Set;
6. **import** javax.persistence.Column;
7. **import** javax.persistence.Entity;
8. **import** javax.persistence.GeneratedValue;
9. **import** javax.persistence.Id;
10. **import** javax.persistence.NamedQueries;
11. **import** javax.persistence.NamedQuery;
12. **import** javax.persistence.OneToMany;
13. **import** javax.persistence.Table;
15. /\*\*
16. \*
17. \* @author Eddie
18. \*/
19. @Entity
20. @Table(name = "CONTACT\_PERSON")
21. @NamedQueries({
22. @NamedQuery(name = ContactPerson.GET\_ALL\_QUERY\_NAME, query = "SELECT c FROM ContactPerson c")})
23. **public** **class** ContactPerson **implements** Serializable {
25. **public** **static** **final** String GET\_ALL\_QUERY\_NAME = "ContactPerson.getAll";
27. **private** **int** conactPersonId;
28. **private** String name;
29. **private** String phoneNumber;
31. **private** Set<Property> properties;
33. **public** ContactPerson() {
34. }
36. **public** ContactPerson(**int** conactPersonId, String name, String phoneNumber) {
37. **this**.conactPersonId = conactPersonId;
38. **this**.name = name;
39. **this**.phoneNumber = phoneNumber;
40. **this**.properties = **new** HashSet<>();
41. }
43. @Id
44. @GeneratedValue
45. @Column(name = "contact\_person\_id")
46. **public** **int** getConactPersonId() {
47. **return** conactPersonId;
48. }
50. **public** **void** setConactPersonId(**int** conactPersonId) {
51. **this**.conactPersonId = conactPersonId;
52. }
53. @Column(name = "name")
54. **public** String getName() {
55. **return** name;
56. }
58. **public** **void** setName(String name) {
59. **this**.name = name;
60. }
62. @Column(name = "phone\_number")
63. **public** String getPhoneNumber() {
64. **return** phoneNumber;
65. }
67. **public** **void** setPhoneNumber(String phoneNumber) {
68. **this**.phoneNumber = phoneNumber;
69. }
71. //enforce the relationship between a property and its contact person using annotation(s). Each property has one and only one contact person. Each contact person might be responsible for zero to many properties
72. @OneToMany(mappedBy = "contactPerson")
73. **public** Set<Property> getProperties() {
74. **return** properties;
75. }
77. **public** **void** setProperties(Set<Property> properties) {
78. **this**.properties = properties;
79. }
81. @Override
82. **public** **int** hashCode() {
83. **int** hash = 7;
84. hash = 53 \* hash + **this**.conactPersonId;
85. **return** hash;
86. }
88. @Override
89. **public** **boolean** equals(Object obj) {
90. **if** (obj == **null**) {
91. **return** **false**;
92. }
93. **if** (getClass() != obj.getClass()) {
94. **return** **false**;
95. }
96. **final** ContactPerson other = (ContactPerson) obj;
97. **if** (**this**.conactPersonId != other.conactPersonId) {
98. **return** **false**;
99. }
100. **return** **true**;
101. }
103. @Override
104. **public** String toString() {
105. **return** **this**.conactPersonId + " - " + name + " - " + phoneNumber;
106. }
107. }

Modified ***Property.java*** file highlighting changes with comments

1. **package** fit5042.tutex.repository.entities;
3. **import** java.io.Serializable;
4. **import** java.util.HashSet;
5. **import** java.util.Set;
7. **import** javax.persistence.CascadeType;
8. **import** javax.persistence.CollectionTable;
9. **import** javax.persistence.Column;
10. **import** javax.persistence.ElementCollection;
11. **import** javax.persistence.Embedded;
12. **import** javax.persistence.Entity;
13. **import** javax.persistence.FetchType;
14. **import** javax.persistence.GeneratedValue;
15. **import** javax.persistence.GenerationType;
16. **import** javax.persistence.Id;
17. **import** javax.persistence.JoinColumn;
18. **import** javax.persistence.ManyToMany;
19. **import** javax.persistence.ManyToOne;
20. **import** javax.persistence.NamedQueries;
21. **import** javax.persistence.NamedQuery;
22. **import** javax.persistence.OneToMany;
23. **import** javax.persistence.OneToOne;
25. /\*\*
26. \*
27. \* @author Eddie
28. \*/
29. @Entity
30. @NamedQueries({
31. @NamedQuery(name = Property.GET\_ALL\_QUERY\_NAME, query = "SELECT p FROM Property p order by p.propertyId desc")})
33. **public** **class** Property **implements** Serializable {
35. **public** **static** **final** String GET\_ALL\_QUERY\_NAME = "Property.getAll";
37. **private** **int** propertyId;
38. **private** **double** size;
39. **private** **int** numberOfBedrooms;
40. **private** **double** price;
42. **private** Address address;
43. **private** ContactPerson contactPerson;
45. **private** Set<String> tags;
47. **public** Property() {
48. **this**.tags = **new** HashSet<>();
49. }
51. **public** Property(**int** propertyId, Address address, **int** numberOfBedrooms, **double** size, **double** price, ContactPerson contactPerson, Set<String> tags) {
52. **this**.propertyId = propertyId;
53. **this**.address = address;
54. **this**.size = size;
55. **this**.numberOfBedrooms = numberOfBedrooms;
56. **this**.price = price;
57. **this**.contactPerson = contactPerson;
58. **this**.tags = tags;
59. }
61. @Id
62. @GeneratedValue(strategy = GenerationType.SEQUENCE)
63. @Column(name = "property\_id")
64. **public** **int** getPropertyId() {
65. **return** propertyId;
66. }
68. **public** **void** setPropertyId(**int** propertyId) {
69. **this**.propertyId = propertyId;
70. }
72. //insert annotation here to make addess as embeded to Property entity and stored as part of Property
73. @Embedded
74. **public** Address getAddress() {
75. **return** address;
76. }
78. **public** **void** setAddress(Address address) {
79. **this**.address = address;
80. }
81. @Column(name = "size")
82. **public** **double** getSize() {
83. **return** size;
84. }
86. **public** **void** setSize(**double** size) {
87. **this**.size = size;
88. }
90. @Column(name = "number\_of\_bedrooms")
91. **public** **int** getNumberOfBedrooms() {
92. **return** numberOfBedrooms;
93. }
95. **public** **void** setNumberOfBedrooms(**int** numberOfBedrooms) {
96. **this**.numberOfBedrooms = numberOfBedrooms;
97. }
99. @Column(name = "price")
100. **public** **double** getPrice() {
101. **return** price;
102. }
104. **public** **void** setPrice(**double** price) {
105. **this**.price = price;
106. }
108. //enforce the relationship between a property and its contact person using annotation(s). Each property has one and only one contact person. Each contact person might be responsible for zero to many properties
109. @ManyToOne
110. **public** ContactPerson getContactPerson() {
111. **return** contactPerson;
112. }
114. **public** **void** setContactPerson(ContactPerson contactPerson) {
115. **this**.contactPerson = contactPerson;
116. }
118. //annotate the attribute tags in Property class so that the tags of a property will be stored in a table called TAG. The tags of a property should be eagerly fetched and the value of each tag must be stored in a column VALUE in the TAG table
119. @ManyToMany(mappedBy = "TAG", fetch = FetchType.EAGER)
120. @CollectionTable(name = "TAG")
121. @Column(name = "VALUE")
122. @ElementCollection
123. **public** Set<String> getTags() {
124. **return** tags;
125. }
127. **public** **void** setTags(Set<String> tags) {
128. **this**.tags = tags;
129. }
131. @Override
132. **public** String toString() {
133. **return** "Property{" + "propertyId=" + propertyId + ", size=" + size + ", numberOfBedrooms=" + numberOfBedrooms + ", price=" + price + ", address=" + address + ", contactPerson=" + contactPerson + ", tags=" + tags + '}';
134. }
135. }

Edit screen

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Search Property by Id

A screenshot of a computer screen

Description automatically generated

Search Property by Budget

A screenshot of a computer

Description automatically generated

{Link to code repository}

<https://github.com/zake111000/FIT5042>

{References}

Word coding highlight

<http://www.planetb.ca/syntax-highlight-word>

How to logging

<https://wiki.eclipse.org/EclipseLink/Examples/JPA/Logging>

Set up derby environment

https://stackoverflow.com/questions/9296941/set-up-derby-home-environment-variable-on-mac

# Declaration

I declare that this eFolio tasks and the linked code are my individual work. I have not copied from any other student’s work or from any other source except where due acknowledgment is made explicitly in the text and code, nor has any part of this submission been written for me by another person.

Signature of student: KeZhang \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_