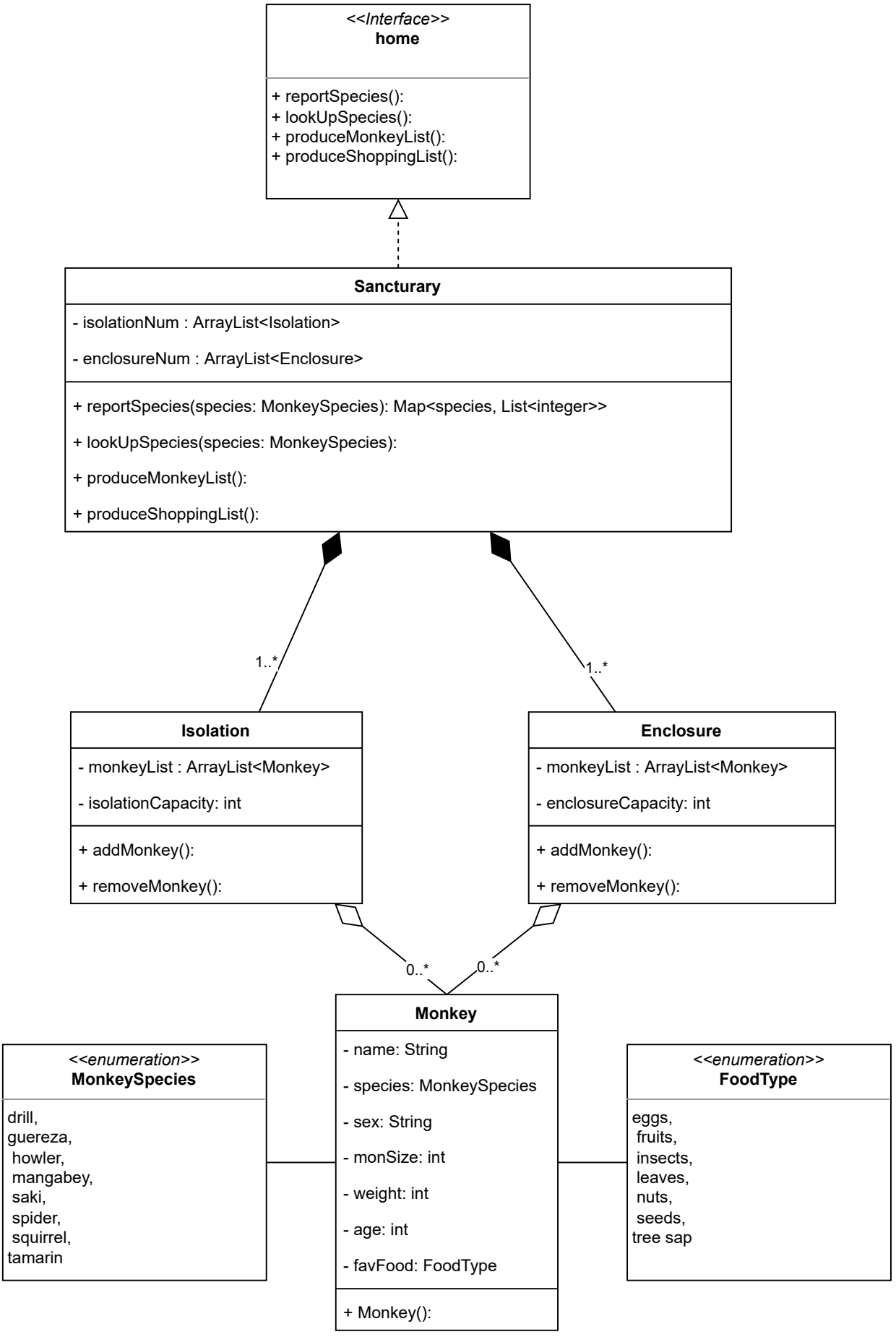
<<Interface>>
home
+ reportSpecies():
+ lookUpSpecies():
+ produceMonkeyList():
+ produceShoppingList():
Sancturary
- isolationNum : ArrayList<Isolation>
- enclosureNum : ArrayList<Enclosure>
+ reportSpecies(species: MonkeySpecies): Map<species, List<integer>>
+ lookUpSpecies(species: MonkeySpecies):
+ produceMonkeyList():
+ produceShoppingList():
1..*
1..*
Isolation
- monkeyList : ArrayList<Monkey>
- isolationCapacity: int
+ addMonkey():
+ removeMonkey():
Enclosure
- monkeyList : ArrayList<Monkey>
- enclosureCapacity: int
+ addMonkey():
+ removeMonkey():
0..*
0..*
Monkey
- name: String
- species: MonkeySpecies
- sex: String
- monSize: int
- weight: int
- age: int
- favFood: FoodType
+ Monkey():
<<enumeration>>
MonkeySpecies
drill,
guereza,
howler,
mangabey,
saki,
spider,
squirrel,
tamarin
<<enumeration>>
FoodType
eggs,
fruits,
insects,
leaves,
nuts,
seeds,
tree sap

MonkeyTest.java

```java
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class MonkeyTest {

  private Monkey defaultMonkey;
  @Before
  public void setUp() {
    defaultMonkey = initialMon("Friday", Species.DRILL,"Male",5,
10,15,"eggs");
  }

  /**
   *  This method is providing short-hand way of creating instances.
   * @param name name
   * @param species species
   * @param sex sex
   * @param monSize monSize
   * @param weight weight
   * @param age age
   * @param favFood favFood
   * @return  short-hand way of creating instances Monkey
   */
  protected Monkey initialMon(String name, Species species, String sex, int
monSize, int weight, int age, String favFood){
    return new Monkey(name,species,sex,monSize,weight,age,favFood);
  }

  @Test
  public void testGetName() {
    String expected = "Friday";
    assertEquals(expected, defaultMonkey.getName());
  }

  @Test
  public void testGetSpecies() {
    String expected = "drill";
    assertEquals(expected, defaultMonkey.getSpecies());
  }

  @Test
  public void testGetSex() {
    String expected = "Male";
    assertEquals(expected, defaultMonkey.getSex());
  }

  @Test
  public void testGetMonSize() {
    int expected = 5;
    assertEquals(expected, defaultMonkey.getMonSize());
  }

  @Test
```

```java
public void testGetWeight() {
  int expected = 10;
  assertEquals(expected, defaultMonkey.getWeight());
}

@Test
public void testGetAge() {
  int expected = 15;
  assertEquals(expected, defaultMonkey.getAge());
}

@Test
public void testGetFavFood() {
  String expected = "eggs";
  assertEquals(expected, defaultMonkey.getFavFood());
}

@Test
public void testSetName() {
  defaultMonkey.setName("Sunday");
  String expected = "Sunday";
  assertEquals(expected, defaultMonkey.getName());
}


@Test
public void testSetSpecies() {
  defaultMonkey.setSpecies(Species.SAKI);
  String expected = String.valueOf(Species.SAKI);
  assertEquals(expected, defaultMonkey.getSpecies());
}

@Test
public void testSetSex() {
  defaultMonkey.setSex("Female");
  String expected = "Female";
  assertEquals(expected, defaultMonkey.getSex());
}

@Test
public void testSetMonSize() {
  defaultMonkey.setMonSize(50);
  int expected = 50;
  assertEquals(expected, defaultMonkey.getMonSize());
}

@Test
public void testSetWeight() {
  defaultMonkey.setWeight(100);
  int expected = 100;
  assertEquals(expected, defaultMonkey.getWeight());
}

@Test
public void testSetAge() {
  defaultMonkey.setAge(150);
  int expected = 150;
```

```
    assertEquals(expected, defaultMonkey.getAge());
  }

  @Test
  public void testSetFavFood() {
    defaultMonkey.setFavFood("nuts");
    String expected = "nuts";
    assertEquals(expected, defaultMonkey.getFavFood());
  }

}
```

```
import org.junit.Before;
import org.junit.Test;

import java.util.ArrayList;

import static org.junit.Assert.*;

public class SanctuaryTest {

  private Home twoIsoThreeEnc;

  @Before
  public void setUp() {
    twoIsoThreeEnc = iso_enc(4,5);
  }

  protected Home iso_enc(ArrayList<Isolation> isolationNum,
ArrayList<Enclosure> enclosuresNum){
    return new Sanctuary(isolationNum, enclosuresNum);
  }

  @Test
  public void testReportSpecies() {
  }

  @Test
  public void testLookUpSpecies() {
  }

  @Test
  public void testProduceMonkeyList() {
  }

  @Test
  public void testProduceShoppingList() {
  }
}
```

```
import project.Food;
import project.Isolation;
import project.Monkey;
import org.junit.Assert;
```

```java
import org.junit.Test;

/**
 * Test cases for Isolation. Verifying that Isolation state is properly
managed, and all Isolation
 * actions are properly validated.
 */
public class IsolationTest {

  @Test
  public void testAddCage() {
    Isolation isolation = new Isolation(10);
    isolation.addCage(5);
    Assert.assertEquals(isolation.getCage(), 15);
  }

  @Test
  public void testGetCage() {
    Isolation isolation = new Isolation(20);
    Assert.assertEquals(isolation.getCage(), 20);
  }

  @Test
  public void testAddMonkey() {
    Isolation isolation = new Isolation(10);
    isolation.addMonkey(new Monkey("spider"));
    isolation.addMonkey(new Monkey("spider"));

    Assert.assertEquals(isolation.getMonkeys().size(), 2);
    isolation.addMonkey(new Monkey("squirrel"));

    Assert.assertEquals(isolation.getMonkeys().size(), 3);
    Assert.assertTrue(isolation.addMonkey(new Monkey("squirrel")));
  }

  @Test
  public void testRemoveMonkey() {
    Isolation isolation = new Isolation(30);
    Monkey m1 = new Monkey("drill");
    Monkey m2 = new Monkey("guereza");
    m1.setAttention("m1", "male", 10, 30, 5, Food.INSECTS);
    m1.setAttention("m2", "male", 10, 20, 5, Food.TREE_SAP);

    Assert.assertTrue(isolation.addMonkey(m1));
    Assert.assertTrue(isolation.addMonkey(m2));
    Assert.assertEquals(isolation.getMonkeys().size(), 2);

    Assert.assertTrue(isolation.removeMonkey(m1));
    Assert.assertEquals(isolation.getMonkeys().size(), 1);
  }
```

```java
public class EnclosureTest {

  @Test
  public void testGetSize() {
```

```
    Enclosure enclosure = new Enclosure("drill", 120);
    Assert.assertEquals(enclosure.getSize(), 120);
  }

  @Test
  public void testGetSpecies() {
    Enclosure enclosure = new Enclosure("guereza", 110);
    Assert.assertEquals(enclosure.getSpecies(), "guereza");
  }

  @Test
  public void testAddMonkey() {
    Enclosure enclosure = new Enclosure("saki", 100);
    Assert.assertFalse(enclosure.addMonkey(new Monkey("saki")));
    Monkey m1 = new Monkey("saki");
    Monkey m2 = new Monkey("squirrel");
    Assert.assertFalse(enclosure.addMonkey(m2));
    m1.setAttention("m1", "male", 10, 10, 10, Food.INSECTS);
    Assert.assertTrue(enclosure.addMonkey(m1));
    Assert.assertEquals(enclosure.getMonkeys().size(), 1);
  }

  @Test
  public void testRemoveMonkey() {
    Enclosure enclosure = new Enclosure("mangabey", 90);
    Monkey m1 = new Monkey("mangabey");
    m1.setAttention("m1", "male", 30, 15, 5, Food.FRUITS);
    Assert.assertTrue(enclosure.addMonkey(m1));
    Assert.assertTrue(enclosure.removeMonkey(m1));
    Assert.assertEquals(enclosure.getMonkeys().size(), 0);
  }
```