|  |  |  |
| --- | --- | --- |
| **CS 1302 – Sample Test 3** | **Name:** | **KEY** |
| No electronics-phone, smart watch, notes, *etc.* allowed. |  | Print: FirstName LastName |

***Write all code using best practices. Reference the handout on last page***

1. Write the abstract *Animal* class including the 2 highlighted methods. You can assume the other methods are implemented.

**public** **abstract** **class** Animal {

 **protected** String name;

 **protected** **int** hunger;

 **public** Animal(String name, **int** hunger) {

 **this**.name = name;

 **this**.hunger = hunger;

 }

 **public** String getName() {

 **return** name;

 }

 **public** **int** getHunger() {

 **return** hunger;

 }

 @Override

 **public** **boolean** equals(Object o) {

 **if**(o **instanceof** Animal) {

 Animal a = (Animal)o;

 **return** **this**.name.equals(a.name);

 }

 **return** **false**;

 }

 **public** **abstract** **int** feed();

}

1. Write the entire *Bear* class.

**public** **class** Bear **extends** Animal {

 **public** Bear(String name, **int** hunger) {

 **super**(name, hunger);

 }

 @Override

 **public** **double** feed() {

 **if** (hunger >= 1) {

 hunger--;

 }

 **return** hunger;

 }

}

1. Write the entire *MommaBear* class.

**public** **class** MommaBear **extends** Bear {

 **private** **int** numCubs;

 **public** MommaBear(String name, **int** hunger, **int** numCubs) {

 **super**(name, hunger);

 **this**.numCubs = numCubs;

 }

 **public** MommaBear(String name, **int** hunger) {

 **this**(name, hunger, 2);

 }

 **public** **int** getNumCubs() {

 **return** numCubs;

 }

 **public** **void** setNumCubs(**int** numCubs) {

 **this**.numCubs = numCubs;

 }

 @Override

 **public** **int** feed() {

 **int** feedAmount = 1 + numCubs;

 **if** (hunger >= feedAmount) {

 hunger-=feedAmount;

 }

 **else** {

 hunger = 0;

 }

 **return** hunger;

 }

}

1. Write the entire *Zoo* class.

**public** **class** Zoo {

 ArrayList<Animal> animals = **new** ArrayList<>(); // part (b)

 **public** **boolean** addAnimal(Animal a) { // part (c)

 **if**(!animals.contains(a)) {

 animals.add(a);

 **return** **true**;

 }

 **return** **false**;

 }

 **public** **void** addAnimal2(Animal a) { // part (d)

 animals.add(a);

 }

 **public** **int** getNumAnimals() { // part (e)

 **return** animals.size();

 }

 **public** Animal getAnimal(**int** i) { // part (f)

 **if**(i>=0 && i<animals.size()) {

 **return** animals.get(i);

 }

 **return** **null**;

 }

 **public** Animal getAnimal(String name) { // part (g)

 Animal key = **new** Bear(name, 0); // dummy/key object

 **int** loc = animals.indexOf(key); // where is dummy in list?

 **return** getAnimal(loc);

 }

 **public** Animal getAnimal2(String name) { // part (g-alt)

 Animal result = **null**;

 **for** (Animal a : animals) {

 **if** (a.getName().equals(name)) {

 result = a;

 **break**;

 }

 }

 **return** result;

 }

 **public** Animal removeAnimal(**int** i) { // part (h)

 **return** animals.remove(i);

 }

 **public** Animal removeAnimal(String name) { // part (i)

 Animal key = **new** Bear(name, 0); // dummy/key object

 **int** loc = animals.indexOf(key); // where is dummy in list?

 **return** removeAnimal(loc);

 }

 **public** Animal removeAnimal2(String name) { // part (i-alt)

 Animal key = getAnimal(name);

 **if** (key != **null**) {

 animals.remove(key);

 **return** key;

 }

 **return** **null**;

 }

 **public** Animal removeAnimal3(String name) { // part (i-alt)

 **int** loc = -1;

 **for** (**int** i = 0; i < animals.size(); i++) {

 Animal a = animals.get(i);

 **if** (a.getName().equals(name)) {

 loc = i;

 **break**;

 }

 }

 **return** removeAnimal(loc);

 }

 **public** ArrayList<MommaBear> getMommaBears(**int** numCubs) { // part (j)

 ArrayList<MommaBear> mommaBears = **new** ArrayList<>();

 **for**(Animal a : animals) {

 **if**(a **instanceof** MommaBear) {

 MommaBear mb = (MommaBear)a;

 **if**(mb.getNumCubs() >= numCubs) {

 mommaBears.add(mb);

 }

 }

 }

 **return** mommaBears;

 }

 **public** **int** feedAll() { // part (k)

 **int** total = 0;

 **for**(Animal a : animals) {

 total += a.feed();

 }

 **return** total;

 }

 @Override

 **public** String toString() {

 String msg = "Animals:\n";

 **for**(Animal a : animals) {

 msg += a + "\n";

 }

 **return** msg;

 }

}

1. Write the following snippets of code:

// (a)

Bear b = **new** Bear("Archie", 5);

// (b)

MommaBear mb1 = **new** MommaBear("Lenoir", 20, 4);

MommaBear mb2 = **new** MommaBear("Sally", 20);

// (c)

Zoo zoo = **new** Zoo();

// (d)

zoo.addAnimal(b);

zoo.addAnimal(mb1);

zoo.addAnimal(mb2);

// (e)

Animal a = zoo.getAnimal(7);

// (f)

Animal a2 = zoo.getAnimal("Urg");

// (g)

Animal a3 = zoo.removeAnimal(4);

// (h)

Animal a4 = zoo.removeAnimal("Quola");

// (i)

ArrayList<MommaBear> mommaBears = zoo.getMommaBears(3);

// (j)

**int** totalHunger = zoo.feedAll();

// (k)

String msg = "";

**for** (**int** i = 0; i < zoo.getNumAnimals(); i++) {

 msg += zoo.getAnimal(i) + "\n";

}

1. Write the following snippets of code:
2. Modify the *Person* class below such that you could sort an *ArrayList<Person>* on *name*.

**public** **class** Person **implements** Comparable<Person>{

 **private** String name;

 **private** **int** age;

 **public** Person(String name, **int** age) {

 **this**.name = name;

 **this**.age = age;

 }

 **public** String getName() {

 **return** name;

 }

 **public** **int** getAge() {

 **return** age;

 }

 @Override

 **public** **int** compareTo(Person p) {

 **return** **this**.name.compareTo(p.name);

 }

 // Alternate version

 **public** **int** compareTo2(Person p) {

 **int** ageDiff = **this**.age-p.age;

 **if**(ageDiff < 0) {

 **return** -1;

 }

 **else** **if**(ageDiff > 0) {

 **return** 1;

 }

 **else** {

 **return** **this**.name.compareTo(p.name);

 }

 }

 @Override

 **public** **boolean** equals(Object o) {

 **if**(o **instanceof** Person) {

 Person p = (Person)o;

 **return** **this**.name.equals(p.name);

 }

 **return** **false**;

 }

 @Override

 **public** String toString() {

 **return** "name=" + name + ", age=" + age + "\n";

 }

}

1. Suppose you have an *ArrayList<Person>* named *persons* for which a number of *Person* objects have been added. Write a line of code to sort this list on *name*.

Collections.*sort*(persons);