AP Java Inheritance Worksheet Find the output of the following program and then answer the True/False questions at the bottom.

Name Kaito Hikino

```
public void setup() {
 Wolf Romulus = new Wolf();
 Wolf Remus = new Wolf();
 Dog ScoobyDoo = new Dog();
 Chihuahua JLosDog =
 new Chihuahua();
 Cat Morris = new Cat();

 Pet[] pets = new Pet[5];
 pets[0] = Romulus;
 pets[1] = Remus;
 pets[2] = ScoobyDoo;
 pets[3] = JLosDog;
 pets[4] = Morris;

 ((Dog)pets[2]).setLicense(1111);
((Dog)pets[3]).setLicense(2222);

 for (int nI = 0;
 nI < pets.length; nI++) {
 System.out.println(
 pets[nI].getSize() + ", " +
pets[nI].speak());
 }
}
class Wolf extends Pet {
 private int myLegs;
 public Wolf() {
 myLegs = 4;
 mySize = 150;
 }
 public int getLegs() {
 return myLegs;
 }
 public String speak() {
 return "Howl!";
 }
}
```

```
class Dog extends Wolf {
 private int myLicense;
 public Dog() {
 mySize = 50;
 }
 public String speak() {
 return "Bark!";
 }
 public void setLicense(int nNumber) {
myLicense = nNumber;
 }
 public int getLicense() {
 return myLicense;
 }
}
class Chihuahua extends Dog {
 public Chihuahua() {
 mySize = 12;
 }
 public String speak() {
 return "Yap!";
 }
}

class Cat extends Pet {
 public String speak() {
 return "Meow";
 }
 public Cat() {
 mySize = 10;
 }
}

class Pet {
 private int mySize;
 public Pet(){mySize = 0;}
 public String speak(){
  return "Pet Sound";}    public int
getSize(){return mySize;}  }
```

The output of the program above is
150, Howl!
150, Howl!
50, Bark!
12, Yap!
10, Meow

True/False Highlight the correct answer

True/False 1. Constructors are never inherited.

True/**False** 2. If you write a method in the derived (sub) class that has the same name, return type and arguments as a method in the base (super) class, you are "overriding" the method of the base (super) class.

True/**False** 3. **`pets[3].getSize() == 0`**

True/**False** 4. **`System.out.println(Romulus.getLegs());`** will cause an exception.

True/False 5. **`System.out.println(pets[1].getLegs());`** will cause an exception.

True/**False** 6. **`System.out.println(ScoobyDoo.getLegs());`** will cause an exception.
True/**False** 7. **`System.out.println(JLosDog.getLegs());`** will cause an exception.
True/False 8. **`System.out.println(Morris.getLegs());`** will cause an exception.
True/**False** 9. **`System.out.println(JLosDog.getLicense());`** will display "1111".
True/False 10. **`System.out.println(Morris.setLicense(3333));`** will cause an exception.
True/**False** 11. The **`Dog`** class overrides the **`Wolf`** class **`getLegs()`** method.
True/False 12. **`Pet[] pets = new Pet[5];`** will cause an exception.
True/False 13. **`Dog pete = new Pet();`** will cause an exception.
True/**False** 14. **`Pet pete = new Dog();`** will cause an exception.
True/False 15. The **`Dog`** class has 3 accessor methods (including inherited methods).
True/**False** 16. The **`Cat`** class has 1 mutator method (including inherited methods).
True/**False** 17. Keeping a **`Wolf`** as a **`Pet`** is a good idea.
True/False 18. **`System.out.println(pets[nI].getSize() + ", " +pets[nI].speak());`** is an example of *polymorphism*.
True/False 19. **`Romulus instance of Pet`**
True/**False** 20. **`Morris instance of Wolf`**