Name: Prathamesh Baban Harad

Overriding

1. Override the taste method from the Candy class in the Chocolate class to return “tastes chocolately”. It should print “tastes sweet!” and then “tastes chocolately”.

```
public class Candy
{
    public String taste()
    {
        return "tastes sweet!"
    }

    public static void main(String[] args)
    {
        Candy c1 = new Candy();
        System.out.println(c1.taste());
        Candy c2 = new Chocolate();
        System.out.println(c2.taste();
    }
}

class Chocolate extends Candy
{
public String taste()
{
System.out.println(super.taste());
```

return "tastes chocolately!";

}

}

2. When a subclass inherits from a superclass, it also inherits its methods; however, it can also override the superclass methods (as well as declare and implement new ones). Consider the following Sports class:

```
class Sports{
    String getName(){
        return "Generic Sports";
    }
    void getNumberOfTeamMembers(){
        System.out.println( "Each team has n players in " + getName() );
    }
}
```

Next, we create a Soccer class that inherits from the Sports class. We can override the getName method and return a different, subclass-specific string:

```
class Soccer extends Sports{
    @Override
    String getName(){
        return "Soccer Class";
    }
}
```

Note: When overriding a method, you should precede it with the @Override annotation. The parameter(s) and return type of an overridden method must be exactly the same as those of the method inherited from the supertype.

Task

Complete the code in your editor by writing an overridden getNumberOfTeamMembers method that prints the same statement as the superclass' getNumberOfTeamMembers method, except that it replaces with (the number of players on a Soccer team).

Output Format

When executed, your completed code should print the following:

Generic Sports
Each team has n players in Generic Sports
Soccer Class
Each team has 11 players in Soccer Class

Solution:

```
class Sports{
String getName(){
return "Generic Sports";
}
void getNumberOfTeamMembers(){
System.out.println( "Each team has n players in " + getName() );
}
}
class Soccer extends Sports{
@Override
String getName(){
return "Soccer Class";
}
void getNumberOfTeamMembers(int n){
System.out.println( "Each team has "+n +" players in " + getName() );
}
}
public class MyClass{
public static void main(String[] args)
{
Sports s1 =new Sports();
System.out.println(s1.getName());
s1.getNumberOfTeamMembers();
Soccer s2 = new Soccer();
System.out.println(s2.getName());
s2.getNumberOfTeamMembers(11);
}
}
```

Overloading:

Write a method that overloads the talk method by taking in a name and printing “Hello” with that name.

```
public class Test1
{
```

```
    public static void talk()
    {
        System.out.println("hello there!");
    }

    public static // FINISH THE METHOD HERE //

    public static void main(String[] args)
    {
        talk("Matthew");
    }
}
```