1. 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

import java.util.\*;

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";

}

// Write your overridden getNumberOfTeamMembers method here

}

public class Solution{

public static void main(String []args){

Sports c1 = new Sports();

Soccer c2 = new Soccer();

System.out.println(c1.getName());

c1.getNumberOfTeamMembers();

System.out.println(c2.getName());

c2.getNumberOfTeamMembers();

}

}

1. When a method in a subclass overrides a method in superclass, it is still possible to call the overridden method using **super** keyword. If you write *super.func()* to call the function *func()*, it will call the method that was defined in the superclass.

You are given a code in the editor. Modify exactly  line so that the code prints the following text:

Hello I am a motorcycle, I am a cycle with an engine.

My ancestor is a cycle who is a vehicle with pedals.

If you lose the original code, click the top left button in the editor and create a new buffer.

import java.util.\*;

import java.io.\*;

class BiCycle

{

String define\_me()

{

return "a vehicle with pedals.";

}

}

class MotorCycle extends BiCycle

{

String define\_me()

{

return "a cycle with an engine.";

}

MotorCycle()

{

System.out.println("Hello I am a motorcycle, I am "+ define\_me());

String temp=define\_me(); ~~Fix me~~

System.out.println("My ancestor is a cycle who is "+ temp );

}

}

class Solution{

public static void main(String []argh)

{

MotorCycle M=new MotorCycle();

}

}