Assignment on Inheritance, Overriding and Abstraction

1. Inheritance

To add a `sing` method to the `Bird` class and modify the main method to print the specified lines, here is the code:class Animal {

```
    void walk() {
        System.out.println("I am walking");
    }
}

class Bird extends Animal {
    void fly() {
        System.out.println("I am flying");
    }

    void sing() {
        System.out.println("I am singing");
    }
}

public class Solution {
    public static void main(String[] args) {
        Bird bird = new Bird();
        bird.walk();
        bird.fly();
        bird.sing();
    }
}
```

2. Abstraction

To create the `MyBook` class that extends the abstract `Book` class, here is the code:

```
abstract class Book {
    String title;
    abstract void setTitle(String s);
    String getTitle() {
        return title;
    }
}

class MyBook extends Book {
    @Override
    void setTitle(String s) {
        title = s;
    }
}

public class Main {
    public static void main(String[] args) {
        MyBook new_novel = new MyBook();
        new_novel.setTitle("A tale of two cities");
        System.out.println("The title is: " + new_novel.getTitle());
    }
}
```

3. Overloading

To override the `getNumberOfTeamMembers` method in the `Soccer` class, here is the code:

```
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";
    }

    @Override
    void getNumberOfTeamMembers() {
        System.out.println("Each team has 11 players in " + getName());
    }
}

public class Main {
    public static void main(String[] args) {
        Sports sport = new Sports();
        System.out.println(sport.getName());
        sport.getNumberOfTeamMembers();

        Soccer soccer = new Soccer();
        System.out.println(soccer.getName());
        soccer.getNumberOfTeamMembers();
    }
}
```

4. Overriding (Duplicate Task)

The fourth task is identical to the third one. Therefore, the same solution applies:

```
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";
    }

    @Override
    void getNumberOfTeamMembers() {
        System.out.println("Each team has 11 players in " + getName());
    }
}

public class Main {
    public static void main(String[] args) {
        Sports sport = new Sports();
        System.out.println(sport.getName());
        sport.getNumberOfTeamMembers();

        Soccer soccer = new Soccer();
        System.out.println(soccer.getName());
        soccer.getNumberOfTeamMembers();
    }
```

}

With these implementations, each task should now meet the specified requirements and produce the expected output.