Can you make Pi?

This problem is inspired by the following picture.

You have been given the completed `Player` class which stores the Player's uniform number (`getNumber()`) and name (`getName()`). (It turns out the Player's name is not relevant in this problem, but I did not realize that fact until I completed this program, and I have no desire to go back and change the problem)
You may assume players numbered 0-9 are listed as a single digit (e.g. the player in the picture is 3, not 03.)

You have been given the completed `TeamLineup` class which stores the target number and an `ArrayList` of all available `Player`s.
You are to complete the static method `getlineup` in the `CanYouMakePi` class. The method has a single `TeamLineup` object as its only parameter. This method will return the longest String possible that matches the beginning of the String `target` in the parameter in the `TeamLineup` constructor.

The `Player` class

```
public class Player {
    private int num;
    private String name;

    // Constructor for objects of class Player
    public Player(int n, String s)     {
        num = n;
        name = s;
    }

    /**
     * @return    the Player's uniform number
     */
    public int getNumber()  { return num;  }

    /**
     * @return    the name of the Player
     */
    public String getName() { return name; }

    public int hashCode()   {
        return new Integer(getNumber()).hashCode() + getName().hashCode();
    }

    public boolean equals(Object obj)  {
        Player p = (Player) obj;
        return getNumber() == p.getNumber() && getName().equals(p.getName());
    }
}
```

The `TeamLineup` class

```
public class TeamLineup
{
    private String teamTarget;
    private ArrayList<Player> team;

    // Constructor for objects of class TeamLineup
    public TeamLineup(String target, ArrayList<Player> players) {
       teamTarget = target;
       team = players;
    }

    public String getTarget() { return teamTarget; }

    public ArrayList<Player> getTeam() { return team; }

    public boolean equals(Object obj) {
        TeamLineup tmp = (TeamLineup) obj;
        return getTarget() == tmp.getTarget() && getTeam().equals(tmp.getTeam());
    }
}
```

You are to complete the static `method getlineup` in the `CanYouMakePi` class. This method will uses the Player numbers from the `Player` Objects stored the `ArrayList` in the `TeamLineup` Object to create the longest `String` possible that matches the beginning of the String `target` from the `TeamLineup` Object.

The following code shows the results of the `getLineUp` method.

The following code	Returns

`ArrayList<Player> team = new ArrayList<Player>();`
`team.add( new Player(3, "player 1"));`
`team.add( new Player(14, "player 12"));`
`team.add( new Player(5, "player 13"));`
 	
`TeamLineup t = new TeamLineup("314159265", team);`	

`CanYouMakePi.getLineUp(t);`	`"314"`

`team.add( 0, new Player(9, "player 15"));`
`team.add( new Player(65, "player 16"));`
`team.add( new Player(15, "player 17"));`
 	
`t = new TeamLineup("314159265", team);`	

`CanYouMakePi.getLineUp(t);`	`"314159"`

`team.add(2,  new Player(2, "player 18"));`
 	
`t = new TeamLineup("314159265", team);`	

`CanYouMakePi.getLineUp(t);`	`"314159265"`

`team.add(2,  new Player(3, "player 183"));`
 	
`t = new TeamLineup("314159265358979", team);`	

`CanYouMakePi.getLineUp(t);`	"31415926535"

Note: This class can be used to match any number as demonstrated in the example on the following page.

The following code shows the results of the `getLineUp` method.

The following code	Returns

`ArrayList<Player> team = new ArrayList<Player>();`
`team.add( new Player(10, "p1"));`
`team.add( new Player(3, "p2"));`
`team.add( new Player(5, "p3"));`
`team.add( new Player(0, "p4"));`
`team.add( new Player(6, "p5"));`
`team.add( new Player(1, "p6"));`
 	
`t = new TeamLineup("6018", team);`	

`CanYouMakePi.getLineUp(t);`	`"601"`

this page intentionally left blank