**WITHOUT GENERICS**

|  |  |  |
| --- | --- | --- |
| **package** com.company;  **public class** Main {   **public static void** main(String[] args) {  Football football = **new** Football(**"football\_1"**);  Baseball baseball = **new** Baseball(**"baseball\_1"**);   Teams teams = **new** Teams(**"team\_1"**);  teams.addPlayer(football);  teams.addPlayer(baseball);   System.***out***.println(teams.numPlayer());  */\*football\_1 is taken by team\_1  baseball\_1 is taken by team\_1  2  \*/*  *//problem is that a team is  // taking all football and  // baseball players, THATS  // WHY WE NEED TO CHECK THE  // PLAYER STATUS   //TO SOLVE THIS PROBLEM  //first we need to put  // different classes of  // TEAMS, although that  // gonna make code DUPLICATION   //else we need GENERICS* } } | **package** com.company;  **public abstract class** Player {  **public** String **name**;   *//constructor* **public** Player(String name) {  **this**.**name** = name;  }   *//getter* **public** String getName() {  **return name**;  } }  FOOTBALL  **package** com.company;  **public class** Football **extends** Player {  **public** Football(String name) {  **super**(name);  } }  BASEBALL class  **package** com.company;  **public class** Baseball **extends** Player {  **public** Baseball(String name) {  **super**(name);  } } | **package** com.company;  **import** java.util.ArrayList;  **public class** Teams {  **private** String **teamName**;  **private int played** = 0;  **private int won** = 0;  **private int lost** = 0;  **private int tied** = 0;   **private** ArrayList<Player> **members** = **new** ArrayList<Player>();   *//set a string name* **public** Teams(String name) {  **this**.**teamName** = name;  }   *//getter* **public** String getTeamName() {  **return teamName**;  }   *//adding player since it  // was an abstract class* **public boolean** addPlayer(Player player){  **if** (**members**.contains(player)) {  System.***out***.println(**"player "** + player.getName() + **" already exists"**);  **return false**;  }  **else**{  **members**.add(player);  System.***out***.println(player.getName() + **" is taken by "** + **this**.**teamName**);  **return true**;  }  }   *//returns the number of  //the player in the team* **public int** numPlayer(){  **return members**.size();  }   *//match result* **public void** matchResult(Teams opponent, **int** ourScore, **int** theirScore){  **if** (ourScore > theirScore)  **won**++;  **else if** (ourScore == theirScore)  **tied**++;  **else  lost**++;  *//getting the opponent result also* **if** (opponent != **null**)  opponent.matchResult(**null**, theirScore, ourScore);  }   *//ranking* **public int** ranking(){  **return** (**won** \* 2) + **tied**;  } } |

**WITH GENERICS**

|  |  |  |
| --- | --- | --- |
| **package** com.company;  **public class** Main {   **public static void** main(String[] args) {  Football football = **new** Football(**"football\_1"**);  Baseball baseball = **new** Baseball(**"baseball\_1"**);   Teams<Football> teams = **new** Teams<Football>(**"team\_1"**);  teams.addPlayer(football);  *// teams.addPlayer(baseball);  // this gonna give an error  // bcz it belongs to base ball class* Teams<Baseball> teams1 = **new** Teams<Baseball>(**"team\_2"**);  teams1.addPlayer(baseball);  */\*  football\_1 is taken by team\_1  baseball\_1 is taken by team\_2  1  1  \*/* System.***out***.println(teams.numPlayer());  System.***out***.println(teams1.numPlayer());   System.***out***.println();   *//it will never give an error here  //but it crashes at the time of build  //reason for the UGLY TYPE CAST at Teams*  Teams<String> teams2 = **new** Teams<String>(**"NOT GONNA WORK"**);  teams2.addPlayer(**"no one"**);  */\*  football\_1 is taken by team\_1  Exception in thread "main" java.lang.ClassCastException:  java.base/java.lang.String cannot be cast to com.company.Player   at com.company.Teams.addPlayer(Teams.java:34)  baseball\_1 is taken by team\_2  at com.company.Main.main(Main.java:23)  1  1  \*/*  } } | **package** com.company;  **public abstract class** Player {  **public** String **name**;   *//constructor* **public** Player(String name) {  **this**.**name** = name;  }   *//getter* **public** String getName() {  **return name**;  } }  FOOTBALL  **package** com.company;  **public class** Football **extends** Player {  **public** Football(String name) {  **super**(name);  } }  BASEBALL class  **package** com.company;  **public class** Baseball **extends** Player {  **public** Baseball(String name) {  **super**(name);  } } | **package** com.company;  **import** java.util.ArrayList;  *//T is called type of* **public class** Teams<T> {  **private** String **teamName**;  **private int played** = 0;  **private int won** = 0;  **private int lost** = 0;  **private int tied** = 0;   **private** ArrayList<T> **members** = **new** ArrayList<T>();   *//set a string name* **public** Teams(String name) {  **this**.**teamName** = name;  }   *//getter* **public** String getTeamName() {  **return teamName**;  }   *//adding player since it  // was an abstract class* **public boolean** addPlayer(T player){  **if** (**members**.contains(player)) {  System.***out***.println(**"player "** + ((Player) player).getName() + **" already exists"**);  **return false**;  }  **else**{  **members**.add(player);  System.***out***.println(((Player) player).getName() + **" is taken by "** + **this**.**teamName**); //UGLY TYPE CASTING  **return true**;  }  }   *//returns the number of  //the player in the team* **public int** numPlayer(){  **return members**.size();  }   *//match result* **public void** matchResult(Teams opponent, **int** ourScore, **int** theirScore){  **if** (ourScore > theirScore)  **won**++;  **else if** (ourScore == theirScore)  **tied**++;  **else  lost**++;  *//getting the opponent result also* **if** (opponent != **null**)  opponent.matchResult(**null**, theirScore, ourScore);  }   *//ranking* **public int** ranking(){  **return** (**won** \* 2) + **tied**;  } } |

**WITH GENERICS SOLVING THE ISSUE : BOUNDED TYPE 🡪 (restricts type)**

|  |  |  |
| --- | --- | --- |
| **package** com.company;  **public class** Main {   **public static void** main(String[] args) {  Football football = **new** Football(**"football\_1"**);  Baseball baseball = **new** Baseball(**"baseball\_1"**);   Teams<Football> teams = **new** Teams<Football>(**"team\_1"**);  teams.addPlayer(football);  Teams<Baseball> teams1 = **new** Teams<Baseball>(**"team\_2"**);  teams1.addPlayer(baseball);  System.***out***.println(teams.numPlayer());  System.***out***.println(teams1.numPlayer());   System.***out***.println();  *//error  //*Teams<String> teams2 = **new** Teams<String>(**"NOT GONNA WORK"**);  //teams2.addPlayer(**"no one"**);  } } | **package** com.company;  **public abstract class** Player {  **public** String **name**;   *//constructor* **public** Player(String name) {  **this**.**name** = name;  }   *//getter* **public** String getName() {  **return name**;  } }  FOOTBALL  **package** com.company;  **public class** Football **extends** Player {  **public** Football(String name) {  **super**(name);  } }  BASEBALL class  **package** com.company;  **public class** Baseball **extends** Player {  **public** Baseball(String name) {  **super**(name);  } } | **package** com.company;  **import** java.util.ArrayList;  ***//Player is the upperbound of T; // we'll now get an error at main  //any class that doesnt derive // from player will get an error***  **public class** Teams<T **extends** Player> {  **private** String **teamName**;  **private int played** = 0;  **private int won** = 0;  **private int lost** = 0;  **private int tied** = 0;   **private** ArrayList<T> **members** = **new** ArrayList<T>();   *//set a string name* **public** Teams(String name) {  **this**.**teamName** = name;  }   *//getter* **public** String getTeamName() {  **return teamName**;  }   *//adding player since it  // was an abstract class* **public boolean** addPlayer(T player){  **if** (**members**.contains(player)) {  *//we dont need the type cast here  //bcz T only extends the Player class* System.***out***.println(**"player "** + player.getName() + **" already exists"**);  **return false**;  }  **else**{  **members**.add(player);  System.***out***.println(player.getName() + **" is taken by "** + **this**.**teamName**);  **return true**;  }  }   *//returns the number of  //the player in the team* **public int** numPlayer(){  **return members**.size();  }   *//match result* **public void** matchResult(Teams opponent, **int** ourScore, **int** theirScore){  **if** (ourScore > theirScore)  **won**++;  **else if** (ourScore == theirScore)  **tied**++;  **else  lost**++;  *//getting the opponent result also* **if** (opponent != **null**)  opponent.matchResult(**null**, theirScore, ourScore);  }   *//ranking* **public int** ranking(){  **return** (**won** \* 2) + **tied**;  } } |