

Workshop 3

Encapsulation & Access Modifiers

1. Create a class **XXXCompetitor** with private fields:
 - o **competitorID** (int), **name** (Name class), **level** (String), **country** (String).
 - o Add getters and setters for all fields.
 - o Create **getOverallScore()** method.
 - o Test by creating objects and updating only allowed fields.

```
1 package Week3Q1;
2
3 public class Name {
4     private String firstName;
5     private String lastName;
6
7     public Name(String firstName, String lastName) {
8         this.firstName = firstName;
9         this.lastName = lastName;
10    }
11
12    public String getFirstName() {
13        return firstName;
14    }
15
16    public void setFirstName(String firstName) {
17        this.firstName = firstName;
18    }
19
20    public String getLastName() {
21        return lastName;
22    }
23
24    public void setLastName(String lastName) {
25        this.lastName = lastName;
26    }
27
28    public String getFullName() {
29        return firstName + " " + lastName;
30    }
31}
32}
```

```
1 package Week3Q1;
2
3 public class XXXCompetitor {
4     private int competitorID;
5     private Name name;
6     private String level;
7     private String country;
8     public XXXCompetitor(int competitorID, Name name, String level, String country) {
9         this.competitorID = competitorID;
0         this.name = name;
1         this.level = level;
2         this.country = country;
3     }
4
5     public int getCompetitorID() {
6         return competitorID;
7     }
8     public void setCompetitorID(int competitorID) {
9         this.competitorID = competitorID;
0     }
10    public Name getName() {
2         return name;
3     }
4     public void setName(Name name) {
5         this.name = name;
6     }
7
8
9     public String getLevel() {
10         return level;
11     }
12     public void setLevel(String level) {
13         this.level = level;
14     }
15     public String getCountry() {
16         return country;
17     }
18     public void setCountry(String country) {
19         this.country = country;
20     }
21     public int getOverallScore() {
22         return 95;
23     }
24 }
```

```

1 package Week3Q1;
2 public class Main {
3     public static void main(String[] args) {
4         Name n1 = new Name("Sneha", "Dahal");
5
6         XXXCompetitor c1 = new XXXCompetitor(101, n1, "Intermediate", "Nepal");
7
8         System.out.println("ID: " + c1.getCompetitorID());
9         System.out.println("Name: " + c1.getName().getFullName());
10        System.out.println("Level: " + c1.getLevel());
11        System.out.println("Country: " + c1.getCountry());
12        System.out.println("Overall Score: " + c1.getOverallScore());
13
14        c1.setLevel("Advanced");
15        c1.getName().setFirstName("S. ");
16
17        System.out.println("\nAfter updates:");
18        System.out.println("Updated Name: " + c1.getName().getFullName());
19        System.out.println("Updated Level: " + c1.getLevel());
20    }
21 }

```

Console <terminated> Main [Java Application] C:\Users\NITRO V15\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.v

ID: 101
Name: Sneha Dahal
Level: Intermediate
Country: Nepal
Overall Score: 95

After updates:
Updated Name: S. Dahal
Updated Level: Advanced

2. Add validation in setters:

- **level** can only be "Beginner", "Intermediate", "Advanced".
- **competitorID** cannot be negative.

```

1 package Week3Q2;
2 public final class Name {
3
4     private final String firstName;
5     private final String lastName;
6
7     public Name(String firstName, String lastName) {
8         this.firstName = firstName;
9         this.lastName = lastName;
10    }
11
12    public String getFirstName() {
13        return firstName;
14    }
15
16    public String getLastName() {
17        return lastName;
18    }
19
20    @Override
21    public String toString() {
22        return firstName + " " + lastName;
23    }
24 }

```

A screenshot of an IDE showing a Java application running. The code in the editor is:

```
1 package Week3Q2;
2 public class Main {
3     public static void main(String[] args) {
4
5         Name n = new Name("Sneha", "Dahal");
6
7         System.out.println(n.getFirstName());
8         System.out.println(n.getLastName());
9
10    }
11 }
```

The console output shows:

```
<terminated> Main (1) [Java Application] C:\Users\NITRO V15\p2\pool\plugin
Sneha
Dahal
```

Immutable Objects

3. Create a **Name** class with **firstName** and **lastName**.

- o Make it immutable (no setters, only final fields + constructor).
- o Test that after object creation, names cannot be changed.

```
1 package Week3Q3;
2 import Week3Q2.Name;
3
4 public class XXXCompetitor {
5     private final int competitorID;
6     private Name name;
7     private String level;
8     private String country;
9     public XXXCompetitor(int competitorID, Name name, String level, String country) {
10         if (competitorID < 0) {
11             throw new IllegalArgumentException("Competitor ID cannot be negative.");
12         }
13         this.competitorID = competitorID;
14         this.name = name;
15         setLevel(level);
16         this.country = country;
17     }
18     public int getCompetitorID() {
19         return competitorID;
20     }
21     public Name getName() {
22         return name;
23     }
24     public String getLevel() {
25         return level;
26     }
27     public String getCountry() {
28         return country;
29     }
30     public void setName(Name name) {
31         this.name = name;
32     }
```

```
1 package Week3Q3;
2
3 import Week3Q2.Name;
4
5 public class BeginnerCompetitor extends XXXCompetitor {
6
7     public BeginnerCompetitor(int id, Name n, String level, String country) {
8         super(id, n, level, country);
9     }
10    @Override
11    public double getOverallScore() {
12        return 50.0;
13    }
14}
15
```

```
1 package Week3Q3;
2
3 import Week3Q2.Name;
4
5 public class IntermediateCompetitor extends XXXCompetitor {
6
7     public IntermediateCompetitor(int id, Name n, String level, String country) {
8         super(id, n, level, country);
9     }
10    @Override
11    public double getOverallScore() {
12        return 75.0;
13    }
14}
15
```

```
1 package Week3Q3;
2
3 import Week3Q2.Name;
4
5 public class AdvancedCompetitor extends XXXCompetitor {
6
7     public AdvancedCompetitor(int id, Name n, String level, String country) {
8         super(id, n, level, country);
9     }
10    @Override
11    public double getOverallScore() {
12        return 95.0;
13    }
14}
15
```

```

1 package Week3Q3;
2
3 import Week3Q2.Name;
4
5 public class Main {
6
7     public static void main(String[] args) {
8
9         XXXCompetitor c = new AdvancedCompetitor(
10             1,
11             new Name("Sneha", "Dahal"),
12             "Advanced",
13             "Nepal"
14         );
15         System.out.println("Polymorphism Score = " + c.getOverallScore());
16         System.out.println("Average Score = " + c.getOverallScore(80, 90, 100));
17         System.out.println("Weighted Score = " + c.getOverallScore(80, 90, 100, 1.2));
18     }
19 }

```

Console X
<terminated> Main (2) [Java Application] C:\Users\NITRO V15\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.
Polymorphism Score = 95.0
Average Score = 90.0
Weighted Score = 108.0

Polymorphism

4. Create subclasses for **XXXCompetitor**:

- o **BeginnerCompetitor, IntermediateCompetitor, AdvancedCompetitor.** Override **getOverallScore()** differently in each subclass.

```

1 package Week3Q4;
2
3 import Week3Q3.*;
4 import Week3Q2.Name;
5
6 public class Main {
7     public static void main(String[] args) {
8         XXXCompetitor c = new AdvancedCompetitor(
9             10,
10            new Name("Snehaa", "Dahal"),
11            "Advanced",
12            "Nepal"
13        );
14         System.out.println("Runtime Polymorphism Score = " + c.getOverallScore());
15     }
16 }

```

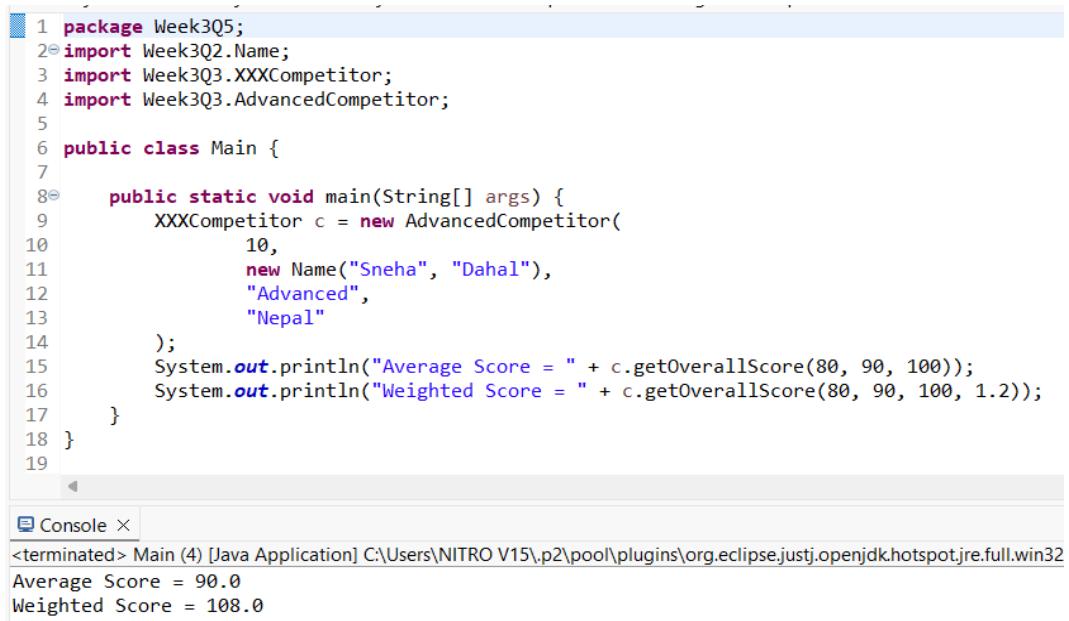
Console X
<terminated> Main (3) [Java Application] C:\Users\NITRO V15\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.w
Runtime Polymorphism Score = 95.0

5. Test runtime polymorphism:

```
XXXCompetitor c = new AdvancedCompetitor(...);  
System.out.println(c.getOverallScore()); // calls subclass version
```

Overload **getOverallScore()** in **XXXCompetitor**:

- One method calculates **average of scores**.
- Another calculates **weighted average** (weight based on level).
- Call both methods from main class.



The screenshot shows an IDE interface with a code editor and a terminal window. The code editor contains a Main.java file with the following content:

```
1 package Week3Q5;  
2 import Week3Q2.Name;  
3 import Week3Q3.XXXCompetitor;  
4 import Week3Q3.AdvancedCompetitor;  
5  
6 public class Main {  
7  
8     public static void main(String[] args) {  
9         XXXCompetitor c = new AdvancedCompetitor(  
10            10,  
11            new Name("Sneha", "Dahal"),  
12            "Advanced",  
13            "Nepal"  
14        );  
15        System.out.println("Average Score = " + c.getOverallScore(80, 90, 100));  
16        System.out.println("Weighted Score = " + c.getOverallScore(80, 90, 100, 1.2));  
17    }  
18 }  
19
```

The terminal window below shows the execution results:

```
Console <terminated> Main (4) [Java Application] C:\Users\NITRO V15\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32  
Average Score = 90.0  
Weighted Score = 108.0
```

Static & Final

6. Make **competitorID** **final** in **XXXCompetitor**.

- Attempt to change it after object creation → show compilation error.

```
1 package Week3Q6;
2
3 public class CompetitionRules {
4     public final void displayRules() {
5         System.out.println("Follow all rules, no cheating, respect judges!");
6     }
7 }
```

```
1 package Week3Q6;
2
3 import Week3Q2.Name;
4 import Week3Q3.XXXCompetitor;
5 import Week3Q3.AdvancedCompetitor;
6
7 public class Main {
8     public static void main(String[] args) {
9         XXXCompetitor c = new AdvancedCompetitor(
10             1,
11             new Name("Sneha", "Dahal"),
12             "Advanced",
13             "Nepal"
14         );
15         System.out.println("Competitor ID = " + c.getCompetitorID());
16         CompetitionRules rules = new CompetitionRules();
17         rules.displayRules();
18     }
19 }
20
```

Console X
<terminated> Main (5) [Java Application] C:\Users\NITRO V15\p2\pool\plugins\org.eclipse.justj.openjdk.Competitor ID = 1
Follow all rules, no cheating, respect judges!

7. Create a **CompetitionRules** class with a **final** method **displayRules()**.

- o Try overriding it in a subclass → explain why it fails.

```
1 package Week3Q7;
2
3 public class Competitor {
4     private int competitorID;
5     private String name;
6     public Competitor(int competitorID, String name) {
7         this.competitorID = competitorID;
8         this.name = name;
9     }
10    public int getCompetitorID() {
11        return competitorID;
12    }
13    public String getName() {
14        return name;
15    }
16    public double getOverallScore() {
17        return 0;
18    }
19 }
```

```

1 package Week3Q7;
2
3 public class ArcheryCompetitor extends Competitor {
4     public ArcheryCompetitor(int id, String name) {
5         super(id, name);
6     }
7
8     @Override
9     public double getOverallScore() {
10        return 85;
11    }
12    public void practiceArchery() {
13        System.out.println(getName() + " is practicing archery!");
14    }
15}
16

1 package Week3Q7;
2
3 public class Main {
4     public static void main(String[] args) {
5         Competitor c = new ArcheryCompetitor(1, "Sneha Dahal");
6         System.out.println("Score via Upcast reference: " + c.getOverallScore());
7         if (c instanceof ArcheryCompetitor) {
8             ((ArcheryCompetitor) c).practiceArchery();
9         }
10    }
11 }

```

Console X
<terminated> Main (6) [Java Application] C:\Users\NITRO V15\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.ful
Score via Upcast reference: 85.0
Sneha Dahal is practicing archery!

Upcasting & Downcasting

8. Upcasting:

```

XXXCompetitor c = new AdvancedCompetitor(...);

c.getOverallScore(); // calls AdvancedCompetitor version

```

Explain why c can only access methods in **XXXCompetitor** (parent class).

```
1 package Week3Q8;
2
3 public class Competitor {
4
5     private int id;
6     private String name;
7
8     public Competitor(int id, String name) {
9         this.id = id;
10        this.name = name;
11    }
12
13     public double getOverallScore() {
14         return 0; // parent default
15     }
16
17     public String getName() {
18         return name;
19     }
20 }
21
```

```
1 package Week3Q8;
2
3 public class ArcheryCompetitor extends Competitor {
4
5     public ArcheryCompetitor(int id, String name) {
6         super(id, name);
7     }
8
9     @Override
10    public double getOverallScore() {
11        return 95.7;
12    }
13    public void shootArrow() {
14        System.out.println(getName() + " is shooting arrows!");
15    }
16 }
17
```

```
1 package Week3Q8;
2
3 public class Main {
4
5     public static void main(String[] args) {
6
7         Competitor c = new ArcheryCompetitor(1, "Sneha");
8
9         System.out.println("Score: " + c.getOverallScore());
10
11         ArcheryCompetitor ac = (ArcheryCompetitor) c;
12         ac.shootArrow();
13     }
14 }
```

Console >
<terminated> Main (7) [Java Application] C:\Users\NITRO V15\.p2\pool\plugins\org.eclipse.jdt.core\src\Week3Q8\Main.java
Score: 95.7
Sneha is shooting arrows!

9. Write a program where you create a superclass **Competitor** and subclass **ArcheryCompetitor**.

- Create an object of **ArcheryCompetitor** and store it in a reference of type **Competitor**.
- Call the **getOverallScore()** method using this upcast reference.
- Try calling a method specific to **ArcheryCompetitor** and observe the result.

```
1 package Week3Q9;
2
3 public class Competitor {
4     private int id;
5     private String name;
6
7     public Competitor(int id, String name) {
8         this.id = id;
9         this.name = name;
10    }
11
12    public double getOverallScore() {
13        return 0; // default
14    }
15
16    public String getName() {
17        return name;
18    }
19 }
20 |
```

```
1 package Week3Q9;
2
3 public class ArcheryCompetitor extends Competitor {
4
5     public ArcheryCompetitor(int id, String name) {
6         super(id, name);
7     }
8
9     @Override
10    public double getOverallScore() {
11        return 88.5;
12    }
13
14    public void shootArrows() {
15        System.out.println(getName() + " is shooting arrows!");
16    }
17 }
18 |
```

The screenshot shows the Eclipse IDE interface. The code editor displays a Java file named Main.java with the following content:

```
1 package Week3Q9;
2
3 public class Main {
4
5     public static void main(String[] args) {
6
7         Competitor c = new ArcheryCompetitor(101, "Sneha Dahal");
8
9         System.out.println("Score = " + c.getOverallScore());
10        ArcheryCompetitor ac = (ArcheryCompetitor) c;
11        ac.shootArrows();
12    }
13 }
14
```

The console window below shows the execution results:

```
Console ×
<terminated> Main [Java Application] C:\Users\NITRO V15\p2\pool\plugins\org.eclipse.justj.open
Score = 88.5
Sneha Dahal is shooting arrows!
```

10. Write a program that defines:

Competitor, ArcheryCompetitor, EsportsCompetitor, and ShootingCompetitor.

- Create one object of each subclass.
- Upcast them all to **Competitor** and store them in an **ArrayList<Competitor>**.
- Iterate through the list and call **getFullDetails()** for each competitor.

The screenshot shows the Eclipse IDE interface. The code editor displays a Java file named Week3Q10.java with the following content:

```
1 package Week3Q10;
2 import java.util.*;
3
4 class Competitor {
5     private int id;
6     private String name;
7
8     public Competitor(int id, String name) {
9         this.id = id;
10        this.name = name;
11    }
12
13     public String getFullDetails() {
14         return "ID: " + id + ", Name: " + name;
15     }
16 }
17
18 class ArcheryCompetitor extends Competitor {
19     public ArcheryCompetitor(int id, String name) {
20         super(id, name);
21     }
22
23     @Override
24     public String getFullDetails() {
25         return super.getFullDetails() + " Category: Archery";
26     }
27 }
28
29 class EsportsCompetitor extends Competitor {
30     public EsportsCompetitor(int id, String name) {
```

```

31         super(id, name);
32     }
33     @Override
34     public String getFullDetails() {
35         return super.getFullDetails() + " Category: Esports";
36     }
37 }
38 class ShootingCompetitor extends Competitor {
39     public ShootingCompetitor(int id, String name) {
40         super(id, name);
41     }
42     @Override
43     public String getFullDetails() {
44         return super.getFullDetails() + " Category: Shooting";
45     }
46 }
47 public class Main {
48     public static void main(String[] args) {
49         Competitor c1 = new ArcheryCompetitor(1, "Kristina");
50         Competitor c2 = new EsportsCompetitor(2, "Sneha");
51         Competitor c3 = new ShootingCompetitor(3, "Krisha");
52         ArrayList<Competitor> list = new ArrayList<>();
53         list.add(c1);
54         list.add(c2);
55         list.add(c3);
56         for (Competitor c : list) {
57             System.out.println(c.getFullDetails());
58         }
59     }
}

```

Console <terminated> Main (8) [Java Application] C:\Users\NITRO V15\p2\pool\plugins\org.eclipse.jus
ID: 1, Name: Kristina Category: Archery
ID: 2, Name: Sneha Category: Esports
ID: 3, Name: Krisha Category: Shooting

11. Downcasting:

Downcast back to **AdvancedCompetitor** to access a **child-specific method**:

((AdvancedCompetitor) c).extraTraining();

```

1 package Week3Q11;
2
3 class Competitor {
4     String name;
5     Competitor(String name) {
6         this.name = name;
7     }
8     void getFullDetails() {
9         System.out.println("Competitor: " + name);
10    }
11 }
12 class AdvancedCompetitor extends Competitor {
13     AdvancedCompetitor(String name) {
14         super(name);
15     }
16     void extraTraining() {
17         System.out.println(name + " is doing extra advanced training.");
18     }
19 }
20 public class Main{
21     public static void main(String[] args) {
22         Competitor c1 = new Competitor("Basic Player");
23         Competitor c2 = new AdvancedCompetitor("Pro Player");
24         Competitor[] list = { c1, c2 };
25         for (Competitor c : list) {
26             c.getFullDetails();
27             if (c instanceof AdvancedCompetitor) {
28                 ((AdvancedCompetitor) c).extraTraining();
29             }
30         }
}

```

Console <terminated> Main (9) [Java Application] C:\Users\NITRO V15\p2\pool\plugins\org.eclipse.justj.openjdk.h
Competitor: Basic Player
Competitor: Pro Player
Pro Player is doing extra advanced training.

12. Use **instanceof** to safely downcast multiple competitor types in a list and call child-specific methods.

```
1 package Week3Q12;
2 import java.util.ArrayList;
3
4 class Competitor {
5     String name;
6     Competitor(String name) {
7         this.name = name;
8     }
9     void getFullDetails() {
10         System.out.println("Competitor: " + name);
11     }
12 }
13 class ArcheryCompetitor extends Competitor {
14     ArcheryCompetitor(String name) {
15         super(name);
16     }
17     void practiceArchery() {
18         System.out.println(name + " is practicing archery.");
19     }
20 }
21 class EsportsCompetitor extends Competitor {
22     EsportsCompetitor(String name) {
23         super(name);
24     }
25     void playGame() {
26         System.out.println(name + " is playing esports games.");
27     }
28 }
29 class ShootingCompetitor extends Competitor {
30     ShootingCompetitor(String name) {
31         super(name);
32     }
33     void practiceShooting() {
34         System.out.println(name + " is practicing shooting.");
35     }
36 }
37 class AdvancedCompetitor extends Competitor {
38     AdvancedCompetitor(String name) {
39         super(name);
40     }
41     void extraTraining() {
42         System.out.println(name + " is doing extra advanced training.");
43     }
44 }
45 public class Main {
46     public static void main(String[] args) {
47         ArrayList<Competitor> competitors = new ArrayList<>();
48         competitors.add(new ArcheryCompetitor("Krisha"));
49         competitors.add(new EsportsCompetitor("Sneha"));
50         competitors.add(new ShootingCompetitor("Shavari"));
51         competitors.add(new AdvancedCompetitor("Prisha"));
52         for (Competitor c : competitors) {
53             c.getFullDetails();
54             if (c instanceof ArcheryCompetitor) {
55                 ((ArcheryCompetitor) c).practiceArchery();
56             } else if (c instanceof EsportsCompetitor) {
57                 ((EsportsCompetitor) c).playGame();
58             } else if (c instanceof ShootingCompetitor) {
59                 ((ShootingCompetitor) c).practiceShooting();
60             } else if (c instanceof AdvancedCompetitor) {
61                 ((AdvancedCompetitor) c).extraTraining();
62             }
63         }
64     }
65 }
```

```
Console ×
<terminated> Main (10) [Java Application] C:\Users\NITRO
Competitor: Krisha
Krisha is practicing archery.
Competitor: Sneha
Sneha is playing esports games.
Competitor: Shavari
Shavari is practicing shooting.
Competitor: Prisha
Prisha is doing extra advanced training.
```

13. Write a program where:

- You upcast an **EsportsCompetitor** object to **Competitor**.
- Then safely downcast it back to **EsportsCompetitor** and call its method **getGamePlayed()**.
- Use **instanceof** to check before downcasting.

The screenshot shows the Eclipse IDE interface with several tabs open at the top, including "Competitor.java", "ArcheryCompetitor.java", "Main.java", "Competitor.java", and "ArcheryCompetitor.java". The main editor window contains the following Java code:

```
1 package Week3Q13;
2
3 class Competitor {
4     String name;
5     Competitor(String name) {
6         this.name = name;
7     }
8     void getFullDetails() {
9         System.out.println("Competitor: " + name);
10    }
11 }
12 class EsportsCompetitor extends Competitor {
13     private int gamesPlayed;
14     EsportsCompetitor(String name, int gamesPlayed) {
15         super(name);
16         this.gamesPlayed = gamesPlayed;
17     }
18     void getGamePlayed() {
19         System.out.println(name + " has played " + gamesPlayed + " games.");
20     }
21 }
22 public class Main {
23     public static void main(String[] args) {
24         Competitor c = new EsportsCompetitor("Sneha", 15);
25         if (c instanceof EsportsCompetitor) {
26             ((EsportsCompetitor) c).getGamePlayed();
27         }
28     }
29 }
```

Below the editor, the "Console" tab is selected, showing the output of the program:

```
<terminated> Main (11) [Java Application] C:\Users\NITRO V15\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jdk11\bin>
Sneha has played 15 games.
```

14. Write a program to demonstrate what happens when you try to downcast a **Competitor** object (that actually refers to an **ArcheryCompetitor**) into an **EsportsCompetitor**.

- Catch and handle the **ClassCastException** properly.

```
1 package Week3Q14;
2
3 class Competitor {
4     String name;
5     Competitor(String name) {
6         this.name = name;
7     }
8     void getFullDetails() {
9         System.out.println("Competitor: " + name);
10    }
11 }
12 class ArcheryCompetitor extends Competitor {
13     ArcheryCompetitor(String name) {
14         super(name);
15     }
16     void practiceArchery() {
17         System.out.println(name + " is practicing archery.");
18     }
19 }
20 class EsportsCompetitor extends Competitor {
21     EsportsCompetitor(String name) {
22         super(name);
23     }
24     void playGame() {
25         System.out.println(name + " is playing esports games.");
26     }
27 }
```

```
28 public class Main {
29     public static void main(String[] args) {
30         Competitor c = new ArcheryCompetitor("Sneha");
31         try {
32             EsportsCompetitor e = (EsportsCompetitor) c;
33             e.playGame();
34         } catch (ClassCastException ex) {
35             System.out.println("Cannot cast ArcheryCompetitor to EsportsCompetitor!");
36         }
37         c.getFullDetails();
38     }
39 }
40
```

Console ×

```
<terminated> Main (12) [Java Application] C:\Users\NITRO V15\p2\pool\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32
Cannot cast ArcheryCompetitor to EsportsCompetitor!
Competitor: Sneha
```