Workshop 3

Encapsulation & Access Modifiers

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

inheritance... | Perimeter.java | protectedke... | Abstraction... | Interfa

```java
package Week3Q1;

public class Name {
    private String firstName;
    private String lastName;

    public Name(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getFullName() {
        return firstName + " " + lastName;
    }
}
```

```java
package Week3Q1;

public class XXXCompetitor {
    private int competitorID;
    private Name name;
    private String level;
    private String country;
    public XXXCompetitor(int competitorID, Name name, String level, String country) {
        this.competitorID = competitorID;
        this.name = name;
        this.level = level;
        this.country = country;
    }

    public int getCompetitorID() {
        return competitorID;
    }
    public void setCompetitorID(int competitorID) {
        this.competitorID = competitorID;
    }
    public Name getName() {
        return name;
    }
    public void setName(Name name) {
        this.name = name;
    }
```

```java
    public String getLevel() {
        return level;
    }
    public void setLevel(String level) {
        this.level = level;
    }
    public String getCountry() {
        return country;
    }
    public void setCountry(String country) {
        this.country = country;
    }
    public int getOverallScore() {
        return 95;
    }
}
```

```java
package Week3Q1;
public class Main {
    public static void main(String[] args) {
        Name n1 = new Name("Sneha", "Dahal");

        XXXCompetitor c1 = new XXXCompetitor(101, n1, "Intermediate", "Nepal");

        System.out.println("ID: " + c1.getCompetitorID());
        System.out.println("Name: " + c1.getName().getFullName());
        System.out.println("Level: " + c1.getLevel());
        System.out.println("Country: " + c1.getCountry());
        System.out.println("Overall Score: " + c1.getOverallScore());

        c1.setLevel("Advanced");
        c1.getName().setFirstName("S. ");

        System.out.println("\nAfter updates:");
        System.out.println("Updated Name: " + c1.getName().getFullName());
        System.out.println("Updated Level: " + c1.getLevel());
    }
}
```

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.

```java
package Week3Q2;
public final class Name {

    private final String firstName;
    private final String lastName;

    public Name(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    @Override
    public String toString() {
        return firstName + " " + lastName;
    }
}
```

```java
package Week3Q2;
public class Main {
    public static void main(String[] args) {

        Name n = new Name("Sneha", "Dahal");

        System.out.println(n.getFirstName());
        System.out.println(n.getLastName());

    }
}
```

Console

<terminated> Main (1) [Java Application] C:\Users\NITRO V15\.p2\pool\plugi

```
Sneha
Dahal
```

Immutable Objects

3. Create a **Name** class with **firstName** and **lastName**.
    - Make it immutable (no setters, only final fields + constructor).
    - Test that after object creation, names cannot be changed.

```java
package Week3Q3;
import Week3Q2.Name;

public class XXXCompetitor {
    private final int competitorID;
    private Name name;
    private String level;
    private String country;
    public XXXCompetitor(int competitorID, Name name, String level, String country) {
        if (competitorID < 0) {
            throw new IllegalArgumentException("Competitor ID cannot be negative.");
        }
        this.competitorID = competitorID;
        this.name = name;
        setLevel(level);
        this.country = country;
    }
    public int getCompetitorID() {
        return competitorID;
    }
    public Name getName() {
        return name;
    }
    public String getLevel() {
        return level;
    }
    public String getCountry() {
        return country;
    }
    public void setName(Name name) {
        this.name = name;
    }
```

```java
package Week3Q3;

import Week3Q2.Name;

public class BeginnerCompetitor extends XXXCompetitor {

    public BeginnerCompetitor(int id, Name n, String level, String country) {
        super(id, n, level, country);
    }
    @Override
    public double getOverallScore() {
        return 50.0;
    }
}
```

```java
package Week3Q3;

import Week3Q2.Name;

public class IntermediateCompetitor extends XXXCompetitor {

    public IntermediateCompetitor(int id, Name n, String level, String country) {
        super(id, n, level, country);
    }

    @Override
    public double getOverallScore() {
        return 75.0;
    }
}
```

```java
package Week3Q3;

import Week3Q2.Name;

public class AdvancedCompetitor extends XXXCompetitor {

    public AdvancedCompetitor(int id, Name n, String level, String country) {
        super(id, n, level, country);
    }

    @Override
    public double getOverallScore() {
        return 95.0;
    }
}
```

```java
package Week3Q3;

import Week3Q2.Name;

public class Main {

    public static void main(String[] args) {

        XXXCompetitor c = new AdvancedCompetitor(
                1,
                new Name("Sneha", "Dahal"),
                "Advanced",
                "Nepal"
        );
        System.out.println("Polymorphism Score = " + c.getOverallScore());
        System.out.println("Average Score = " + c.getOverallScore(80, 90, 100));
        System.out.println("Weighted Score = " + c.getOverallScore(80, 90, 100, 1.2));
    }
}
```

```
Console ×
<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**:

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

```java
package Week3Q4;

import Week3Q3.*;
import Week3Q2.Name;

public class Main {
    public static void main(String[] args) {
        XXXCompetitor c = new AdvancedCompetitor(
                10,
                new Name("Snehaa", "Dahal"),
                "Advanced",
                "Nepal"
        );
        System.out.println("Runtime Polymorphism Score = " + c.getOverallScore());
    }
}
```

```
Console ×
<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.

```java
package Week3Q5;
import Week3Q2.Name;
import Week3Q3.XXXCompetitor;
import Week3Q3.AdvancedCompetitor;

public class Main {

    public static void main(String[] args) {
        XXXCompetitor c = new AdvancedCompetitor(
                10,
                new Name("Sneha", "Dahal"),
                "Advanced",
                "Nepal"
        );
        System.out.println("Average Score = " + c.getOverallScore(80, 90, 100));
        System.out.println("Weighted Score = " + c.getOverallScore(80, 90, 100, 1.2));
    }
}

```

```
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.

```java
package Week3Q6;

public class CompetitonRules {
    public final void displayRules() {
        System.out.println("Follow all rules, no cheating, respect judges!");
    }
}
```

```java
package Week3Q6;

import Week3Q2.Name;
import Week3Q3.XXXCompetitor;
import Week3Q3.AdvancedCompetitor;

public class Main {
    public static void main(String[] args) {
        XXXCompetitor c = new AdvancedCompetitor(
                1,
                new Name("Sneha", "Dahal"),
                "Advanced",
                "Nepal"
        );
        System.out.println("Competitor ID = " + c.getCompetitorID());
        CompetitonRules rules = new CompetitonRules();
        rules.displayRules();
    }
}
```

Console

```
<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()**.

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

```java
package Week3Q7;

public class Competitor {
    private int competitorID;
    private String name;
    public Competitor(int competitorID, String name) {
        this.competitorID = competitorID;
        this.name = name;
    }
    public int getCompetitorID() {
        return competitorID;
    }
    public String getName() {
        return name;
    }
    public double getOverallScore() {
        return 0;
    }
}
```

```java
package Week3Q7;

public class ArcheryCompetitor extends Competitor {
    public ArcheryCompetitor(int id, String name) {
        super(id, name);
    }

    @Override
    public double getOverallScore() {
        return 85;
    }
    public void practiceArchery() {
        System.out.println(getName() + " is practicing archery!");
    }
}

```

```java
package Week3Q7;

public class Main {
    public static void main(String[] args) {
        Competitor c = new ArcheryCompetitor(1, "Sneha Dahal");
        System.out.println("Score via Upcast reference: " + c.getOverallScore());
        if (c instanceof ArcheryCompetitor) {
            ((ArcheryCompetitor) c).practiceArchery();
        }
    }
}
```

```
Console ×
<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).

```java
package Week3Q8;

public class Competitor {

    private int id;
    private String name;

    public Competitor(int id, String name) {
        this.id = id;
        this.name = name;
    }

    public double getOverallScore() {
        return 0;   // parent default
    }

    public String getName() {
        return name;
    }
}
```

```java
package Week3Q8;

public class ArcheryCompetitor extends Competitor {

    public ArcheryCompetitor(int id, String name) {
        super(id, name);
    }

    @Override
    public double getOverallScore() {
        return 95.7;
    }
    public void shootArrow() {
        System.out.println(getName() + " is shooting arrows!");
    }
}
```

```java
package Week3Q8;

public class Main {

    public static void main(String[] args) {

        Competitor c = new ArcheryCompetitor(1, "Sneha");

        System.out.println("Score: " + c.getOverallScore());

        ArcheryCompetitor ac = (ArcheryCompetitor) c;
        ac.shootArrow();
    }
}
```

Console ×

```
<terminated> Main (7) [Java Application] C:\Users\NITRO V15\.p2\pool\plugins\org.eclip
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.

```java
package Week3Q9;

public class Competitor {
    private int id;
    private String name;

    public Competitor(int id, String name) {
        this.id = id;
        this.name = name;
    }

    public double getOverallScore() {
        return 0; // default
    }

    public String getName() {
        return name;
    }
}
```

```java
package Week3Q9;

public class ArcheryCompetitor extends Competitor {

    public ArcheryCompetitor(int id, String name) {
        super(id, name);
    }

    @Override
    public double getOverallScore() {
        return 88.5;
    }

    public void shootArrows() {
        System.out.println(getName() + " is shooting arrows!");
    }
}
```

```java
package Week3Q9;

public class Main {

    public static void main(String[] args) {

        Competitor c = new ArcheryCompetitor(101, "Sneha Dahal");

        System.out.println("Score = " + c.getOverallScore());
        ArcheryCompetitor ac = (ArcheryCompetitor) c;
        ac.shootArrows();
    }
}
```

```
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.

```java
package Week3Q10;
import java.util.*;

class Competitor {
    private int id;
    private String name;

    public Competitor(int id, String name) {
        this.id = id;
        this.name = name;
    }

    public String getFullDetails() {
        return "ID: " + id + ", Name: " + name;
    }
}

class ArcheryCompetitor extends Competitor {
    public ArcheryCompetitor(int id, String name) {
        super(id, name);
    }

    @Override
    public String getFullDetails() {
        return super.getFullDetails() + " Category: Archery";
    }
}

class EsportsCompetitor extends Competitor {
    public EsportsCompetitor(int id, String name) {
```

```
        super(id, name);
    }
    @Override
    public String getFullDetails() {
        return super.getFullDetails() + " Category: Esports";
    }
}
class ShootingCompetitor extends Competitor {
    public ShootingCompetitor(int id, String name) {
        super(id, name);
    }
    @Override
    public String getFullDetails() {
        return super.getFullDetails() + " Category: Shooting";
    }
}
public class Main {
    public static void main(String[] args) {
        Competitor c1 = new ArcheryCompetitor(1, "Kristina");
        Competitor c2 = new EsportsCompetitor(2, "Sneha");
        Competitor c3 = new ShootingCompetitor(3, "Krisha");
        ArrayList<Competitor> list = new ArrayList<>();
        list.add(c1);
        list.add(c2);
        list.add(c3);
        for (Competitor c : list) {
            System.out.println(c.getFullDetails());
        }
    }
```

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();

```
package Week3Q11;

class Competitor {
    String name;
    Competitor(String name) {
        this.name = name;
    }
    void getFullDetails() {
        System.out.println("Competitor: " + name);
    }
}
class AdvancedCompetitor extends Competitor {
    AdvancedCompetitor(String name) {
        super(name);
    }
    void extraTraining() {
        System.out.println(name + " is doing extra advanced training.");
    }
}
public class Main{
    public static void main(String[] args) {
        Competitor c1 = new Competitor("Basic Player");
        Competitor c2 = new AdvancedCompetitor("Pro Player");
        Competitor[] list = { c1, c2 };
        for (Competitor c : list) {
            c.getFullDetails();
            if (c instanceof AdvancedCompetitor) {
                ((AdvancedCompetitor) c).extraTraining();
            }
```

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.

```java
package Week3Q12;
import java.util.ArrayList;

class Competitor {
    String name;
    Competitor(String name) {
        this.name = name;
    }
    void getFullDetails() {
        System.out.println("Competitor: " + name);
    }
}
class ArcheryCompetitor extends Competitor {
    ArcheryCompetitor(String name) {
        super(name);
    }
    void practiceArchery() {
        System.out.println(name + " is practicing archery.");
    }
}
class EsportsCompetitor extends Competitor {
    EsportsCompetitor(String name) {
        super(name);
    }
    void playGame() {
        System.out.println(name + " is playing esports games.");
    }
}
class ShootingCompetitor extends Competitor {

    ShootingCompetitor(String name) {
        super(name);
    }
    void practiceShooting() {
        System.out.println(name + " is practicing shooting.");
    }
}
class AdvancedCompetitor extends Competitor {
    AdvancedCompetitor(String name) {
        super(name);
    }
    void extraTraining() {
        System.out.println(name + " is doing extra advanced training.");
    }
}
public class Main {
    public static void main(String[] args) {
        ArrayList<Competitor> competitors = new ArrayList<>();
        competitors.add(new ArcheryCompetitor("Krisha"));
        competitors.add(new EsportsCompetitor("Sneha"));
        competitors.add(new ShootingCompetitor("Shavari"));
        competitors.add(new AdvancedCompetitor("Prisha"));
        for (Competitor c : competitors) {
            c.getFullDetails();
            if (c instanceof ArcheryCompetitor) {
                ((ArcheryCompetitor) c).practiceArchery();
            } else if (c instanceof EsportsCompetitor) {
                ((EsportsCompetitor) c).playGame();
            } else if (c instanceof ShootingCompetitor) {
                ((ShootingCompetitor) c).practiceShooting();
            } else if (c instanceof AdvancedCompetitor) {
                ((AdvancedCompetitor) c).extraTraining();
            }
        }
    }
}
```

```
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.

```java
package Week3Q13;

class Competitor {
    String name;
    Competitor(String name) {
        this.name = name;
    }
    void getFullDetails() {
        System.out.println("Competitor: " + name);
    }
}
class EsportsCompetitor extends Competitor {
    private int gamesPlayed;
    EsportsCompetitor(String name, int gamesPlayed) {
        super(name);
        this.gamesPlayed = gamesPlayed;
    }
    void getGamePlayed() {
        System.out.println(name + " has played " + gamesPlayed + " games.");
    }
}
public class Main {
    public static void main(String[] args) {
        Competitor c = new EsportsCompetitor("Sneha", 15);
        if (c instanceof EsportsCompetitor) {
            ((EsportsCompetitor) c).getGamePlayed();
        }
    }
}
```

```
Console ×
<terminated> Main (11) [Java Application] C:\Users\NITRO V15\.p2\pool\plugins\org.eclipse.justj.openjdk.hotsp
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.

```java
package Week3Q14;

class Competitor {
    String name;
    Competitor(String name) {
        this.name = name;
    }
    void getFullDetails() {
        System.out.println("Competitor: " + name);
    }
}
class ArcheryCompetitor extends Competitor {
    ArcheryCompetitor(String name) {
        super(name);
    }
    void practiceArchery() {
        System.out.println(name + " is practicing archery.");
    }
}
class EsportsCompetitor extends Competitor {
    EsportsCompetitor(String name) {
        super(name);
    }
    void playGame() {
        System.out.println(name + " is playing esports games.");
    }
}
public class Main {
    public static void main(String[] args) {
        Competitor c = new ArcheryCompetitor("Sneha");
        try {
            EsportsCompetitor e = (EsportsCompetitor) c;
            e.playGame();
        } catch (ClassCastException ex) {
            System.out.println("Cannot cast ArcheryCompetitor to EsportsCompetitor!");
        }
        c.getFullDetails();
    }
}
```

Console ×

<terminated> Main (12) [Java Application] C:\Users\NITRO V15\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win3

```
Cannot cast ArcheryCompetitor to EsportsCompetitor!
Competitor: Sneha
```