Sure! Here's a quick overview of each topic with simple Java programs and explanations.

1. Access Modifiers (Protected Example)

Let's look at an example of protected access. This allows members to be accessed in the same package and in subclasses.

```
// File: Animal.java
package animals;

public class Animal {
    protected String name = "Unknown Animal";

    protected void displayName() {
        System.out.println("Animal: " + name);
    }
}

// File: Dog.java (subclass in the same package)
package animals;

public class Dog extends Animal {
    public void setName(String name) {
        this.name = name;  // Accessing protected field
    }

    public void showAnimal() {
        displayName();  // Accessing protected method
    }
}

// File: Main.java (in the same package)
package animals;

public class Main {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.setName("Bulldog");
        dog.showAnimal();  // Displays "Animal: Bulldog"
    }
}
```

Explanation:

name and displayName() in the Animal class are protected.

The Dog class, which extends Animal, can directly access and modify name and call displayName().

(for more understanding, read another brief material in your laptop)

2. Thread Safety

Let’s see an example of making a method thread-safe using synchronized.

```
public class Counter {
    private int count = 0;

    // Synchronized method to make it thread-safe
    public synchronized void increment() {
        count++;
    }

    public int getCount() {
        return count;
    }

    public static void main(String[] args) {
        Counter counter = new Counter();

        // Create multiple threads to increment the counter
        Thread t1 = new Thread(() -> {
            for (int i = 0; i < 1000; i++) counter.increment();
        });

        Thread t2 = new Thread(() -> {
            for (int i = 0; i < 1000; i++) counter.increment();
        });

        t1.start();
        t2.start();

        // Wait for both threads to finish
        try {
            t1.join();
            t2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("Final Count: " + counter.getCount()); // Should be 2000
    }
}
```

Explanation:

The increment() method is marked as synchronized, ensuring only one thread can access it at a time.

Without synchronization, both threads could modify count simultaneously, causing data

inconsistency.

Here, both threads increase the count to 2000, showing thread safety.

3. StringBuilder (Non-Thread Safe) vs. StringBuffer (Thread Safe)

Here’s an example showing StringBuilder and StringBuffer.

```
public class StringExample {
    public static void main(String[] args) {
        // StringBuilder (non-thread-safe, faster in single-threaded cases)
        StringBuilder sb = new StringBuilder();
        sb.append("Hello ");
        sb.append("World");
        System.out.println("StringBuilder result: " + sb.toString());

        // StringBuffer (thread-safe, slower due to synchronization)
        StringBuffer sbf = new StringBuffer();
        sbf.append("Hello ");
        sbf.append("World");
        System.out.println("StringBuffer result: " + sbf.toString());
    }
}
```

Explanation:

StringBuilder: Faster but not thread-safe. It’s ideal in single-threaded environments.

StringBuffer: Thread-safe because its methods are synchronized, making it slower. Use it if you need thread safety in multi-threaded applications