Atomic Reference

```java
import java.util.function.UnaryOperator;
public class AtomicReference<T> {
	private T value;
	private Object lock;

	public AtomicReference() {
		this(null);
	}
	public AtomicReference(T value) {
		this.value = value;
		this.lock = new Object();
	}
	public T get() {
		synchronized (lock) {
			return value;
		}
	}
	public void set(T value) {
		synchronized (lock) {
			this.value = value;
		}
	}
	public T getAndSet(T value) {
		synchronized (lock) {
			T result = this.value;
			this.value = value;
			return result;
		}
	}
	public T getAndUpdate(UnaryOperator<T> update) {
		synchronized (lock) {
			T result = this.value;
			this.value = update.apply(value);
			return result;
		}
	}
	public T updateAndGet(UnaryOperator<T> update) {
		synchronized (lock) {
			T result = update.apply(value);
			this.value = result;
			return result;
		}
	}
}
```

```java
public class AtomicReferenceExample {

    public static void main(String[] args) {
        new AtomicReferenceExample().go();
    }

    private void go() {
        AtomicReference<Integer> counter = new AtomicReference<Integer>(1);
        int i = counter.get();

        Incrementer incrementer = new Incrementer();

        while(i <= 10) {
            System.out.println(i);
            i = counter.updateAndGet(incrementer);
        }
    }

    private static class Incrementer implements UnaryOperator<Integer> {
        @Override
        public Integer apply(Integer value) {
            return value + 1;
        }
    }
}
```