Java Functional Interfaces provide a way to leverage the power of functional programming in Java. They are interfaces that have only one abstract method and can be used as lambda expressions or method references. Functional interfaces make it easier to work with functional programming constructs like lambdas and streams. Here are a couple of advantages of Java Functional Interfaces along with code examples:

1. **Lambdas and Method References:** Functional interfaces allow you to use lambda expressions or method references to provide concise and more readable implementations for the single abstract method. This leads to more expressive and compact code.

   Example 1: Using a Functional Interface with a Lambda Expression
   ```java
   // Functional Interface
   @FunctionalInterface
   interface MyFunction {
       int apply(int a, int b);
   }

   public static void main(String[] args) {
       MyFunction add = (a, b) -> a + b;
       MyFunction subtract = (a, b) -> a - b;

       System.out.println(add.apply(5, 3));      // Output: 8
       System.out.println(subtract.apply(5, 3)); // Output: 2
   }
   ```

2. **Method Composition:** Functional interfaces can be composed or combined to create more complex functionality. This allows you to chain multiple functions together, making your code modular and maintainable.

   Example 2: Combining Functional Interfaces
   ```java
   // Functional Interfaces
   @FunctionalInterface
   interface MyFunction {
       int apply(int a, int b);
   }

   @FunctionalInterface
   interface IntUnaryOperator {
       int apply(int a);
   }
```

```
    public static void main(String[] args) {
        MyFunction add = (a, b) -> a + b;
        IntUnaryOperator doubleIt = a -> a * 2;

        // Compose functions
        IntUnaryOperator addAndDouble = a -> doubleIt.apply(add.apply(a, 3));

        System.out.println(addAndDouble.apply(5)); // Output: 16 (5 + 3 = 8, 8 * 2 = 16)
    }
```

3. **Interoperability:** Functional interfaces facilitate interoperability with older code that uses interfaces with a single abstract method. This makes it easier to integrate modern functional constructs with existing Java code.

These advantages of functional interfaces make Java more expressive, maintainable, and adaptable to modern programming paradigms. They are particularly useful when working with collections, parallel processing, and asynchronous programming, as they allow you to pass behavior as data.