a.

1-Imperative programming focuses on "how" while functional programming is declarative, which concentrates on "what". The functional programming makes code more self-documenting-the sequence of function calls mirrors precisely the requirements.

2-Imperative programming cannot guarantee to return the same result for two calls while functional programming can do that by making the functions referential transparency. Unlike the imperative programming, functions in functional programming do not cause a change of state also known as modification of their environment.

b.

Declarative programming is a programming paradigm—a style of building the structure and elements of computer programs—that expresses the logic of a computation without describing its control flow. This programming style attempts to minimize or eliminate side effects by describing what the program must accomplish in terms of the problem domain, rather than describe how to accomplish it as a sequence of the programming language primitives. For example, programming by database query language such as SQL.

c.

Functional interface is an interface that requires exactly one abstract method to be implemented in order to satisfy the requirements of the interface. An implementation of a functional interface is called a functor which is a class with just one implemented function, implementing the interface. A functor is a functional object that acts like a function. A closure is a functor embedded inside another class. Operationally, a closure is a record storing a function together with an environment.

The Callable<T> interface is a functional interface:

public interface Callable<V> {

V call() throws Exception;

}

public static class WordLengthCallable

implements Callable {

private String word;

public WordLengthCallable(String word) {

this.word = word;

}

public Integer call() {

return Integer.valueOf(word.length());

}

}

The static class WordLengthCallable is a functor.

In Java 7, inner class including anonymous inner class is closure.

d.

Benefits with functional style programming in Java:

1-Programs are more compact, easier to write, and easier to read/understand

2-Easier to demonstrate correctness of functional programs

3-Easier to test; less likely that a test of a subroutine will fail tomorrow if it passed today since there are no side effects

e.

i. λx.x+2x2

ii. λxy.y–x+xy

iii. λxyz.z–(x+y)

f.

i. no parameter, s and t are free variables

ii. u and v are parameters, a, b and x are free variables

iii. s and t are parameters, ignoreCase is free variable

g.

public class MyClass {

public boolean myMethod(MyClass cl) {

Function<MyClass,Boolean> f = this::equals;

return f.apply(cl);

}

}

h.

i. x -> System.out.println(x)

ii. It is the type of object::instanceMethod because the println is the instance method of PrintStream class, and System.out is an object of this class.

j.

i. () -> Math.random()

ii.

public class MyClass {

public static void main(String[] args) {

Supplier<Double> f = () -> Math.random();

System.out.println(f.get());

}

}

iii.

public class Main {

public static void main(String[] args) {

class MySupplier implements Supplier<Double> {

@Override

public Double get() {

return Math.random();

}

}

MySupplier ms = new MySupplier();

System.out.println(ms.get());

}

}