ИНТЕРФЕЙСЫ И ЛЯМБДА-ВЫРАЖЕНИЯ

Функциональные интерфейсы

- Интерфейс с единственным методом

```
public interface Comparator <T>{
    int compare(T first, T second);
}
public interface Runnable {
    void run();
}
public interface ActionListener {
    void ActionPerformed (ActionEvent e);
}
```

Функциональные интерфейсы

- Аннотация

  @FunctionalInterface

  Чаще всего для реализации интерфейса используют анонимные классы

Без использования лямбда-выражения

```
String [ ] words;
. . .
Arrays.sort (words); // стандартное сравнение
Arrays.sort(words, new Comparator<String>(){
  public int compare(String first, String second){
            return first.length() – second.length();
}});
```

Лямбда-выражение

Java 8 внесено расширение на уровне языка. Можно использовать там, где ожидается объект класса, реализующего функциональный интерфейс.

```
String [ ] words;
. . .
Arrays.sort (words,
      (first,second)-> first.length() – second.length());

Comparator<String> comp =
      (first,second)-> first.length() – second.length();

Arrays.sort (words, comp );
```

Лямбда-выражение

- Лямбда-выражение можно присваивать переменной, типом которой является функциональный интерфейс
- Лямбда-выражение можно использовать там, где требуется объект класса, реализующего функциональный интерфейс.

Формат записи

```
Runnable runn = () -> System.out.println(“!!!!!”);
ActionListener lsnr =
        event ->System.out.println(“BUTTON”);
Runnable mult = () -> {
        System.out.println(“!!!!!”);
        for ( ; ;) System.out.println(“.”);
}
BinaryOperator <Long> add = (x,y) -> x+y;
```

Использование значений

При реализации анонимным классом захват контекста – финальные переменные

```
final String name = getUserName();
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        System.out.println("Hi "+name);
    }
});
```

Использование значений

В лямбда-выражении – эффективно финальные переменные (становятся такими, если попадают в лямбда-выражение)

```
String name = getUserName();
//не допустимо name = getOtherName();
button.addActionListener( event ->
                System.out.println("Hi "+name);
);
```

Функциональные интерфейсы

- Пакет

  java.util.function,*

Интерфейс	Метод
Predicat <T>	boolean test (T t)
BinaryOperator<T>	T apply (T f, T s)
Consumer <T>	void accept (T t)
Supplier <T>	T get()

Ссылки на методы

- В лямбда-выражении можно использовать имеющиеся методы классов

```
Arrays.sort(words,(x,y)->x.compareToIgnoreCase(y));
```

- Для упрощения можно использовать ссылку на метод класса String

```
Arrays.sort(words,String::compareToIgnoreCase);
```

Ссылки на методы

```
List <String> ws =Arrays.asList(words);
. . .
// forEach – параметр Consumer<? super T>

ws.forEach(x->System.out.println(x));
ws.forEach(System.out::println);

// replaceAll –параметр UnaryOperator<E>
ws.replaceAll(x->x.toUpperCase());
ws.replaceAll(String::toUpperCase());


// removeIf – параметр Predicat <?super E>
ws.removeIf(x->x.isNull());
ws.removeIf(Objects::isNull);
```