

Java Interface

Regular meeting for get knowledge and tech in
EDUC

Что это?

- **Интерфейс** ([англ. interface](#)) — программная/синтаксическая структура, определяющая отношение между объектами, которые разделяют определённое поведенческое множество и не связаны никак иначе. При проектировании классов, разработка интерфейса тождественна разработке спецификации (множества методов, которые каждый класс, использующий интерфейс, *должен* реализовывать).

Проще говоря

- Интерфейс – это контракт, в рамках которого части программы, зачастую написанные разными людьми, взаимодействуют между собой и со внешними приложениями. Интерфейсы работают со слоями сервисов, безопасности, DAO и т.д. Это позволяет создавать модульные конструкции, в которых для изменения одного элемента не нужно трогать остальные.
- Ключевое слово `interface` используется для создания полностью абстрактных классов. Создатель интерфейса определяет имена методов, списки аргументов и типы возвращаемых значений, но не тела методов.

Подробности

- Интерфейс может содержать:
 - Поля – константы (все поля интерфейса автоматически являются `public final static`, модификаторы указывать необязательно)
 - Методы (методы интерфейса являются `public abstract`)
 - Начиная с `java 8` – имплементацию методов по умолчанию (ключевое слово `default`)
- Интерфейс не может содержать:
 - Поля - переменные
- Так как интерфейсы являются абстрактными классами нельзя напрямую создать объект интерфейса

Пример


```
public interface Budget {  
  
    String Currency = "Рубль"; // поле - константа  
  
    String getCategory(); // объявление метода  
  
    default String getDescription() { // метод по умолчанию  
        |     return "Описание";  
    }  
  
}
```

Реализация

- Для реализации интерфейса он должен быть указан при декларации класса с помощью ключевого слова `implements`. Пример:

```
public class BudgetImpl implements Budget { // ключевое слово implements

    private String category; // поле - переменная

    public String getCategory() { // имплементация метода getCategory
    |     return category;
    |
    | }
    |
    | 
    | public void setCategory(String category) // классы могут содержать методы не указанные в интерфейсе
    | {
    | |     this.category = category;
    | |
    | | }
    | }
    }
```

Расширение и множественная реализация

- Интерфейс может наследоваться от другого интерфейса через ключевое слово `extends`
- Интерфейсов у класса может быть несколько, тогда они перечисляются за ключевым словом `implements` и разделяются запятыми.

```
public interface PlannedBuget extends Budget{  
    String getCronMask();  
}
```

```
public class PlannedBudgetImpl extends BudgetImpl implements Budget, PlannedBuget {  
    String CronMask;  
  
    public String getCronMask() {  
        return CronMask;  
    }  
  
    public void setCronMask(String cronMask) {  
        this.CronMask = cronMask;  
    }  
}
```

Примеры использования

```
class EntityManager {  
    public void saveRecord(Budget budget) { // интерфейсы можно указывать в качестве параметров методов  
    }  
}
```

```
Budget budget1 = new BudgetImpl(); // можно объявлять переменную с типом интерфейс и создавать конкретную реализацию  
BudgetImpl budget2 = new BudgetImpl();  
Budget budget3 = new PlannedBudgetImpl();  
PlannedBudget budget4 = new PlannedBudgetImpl();
```

```
EntityManager entityManager = new EntityManager();  
entityManager.saveRecord(budget1); // в метод можно передавать любую реализацию интерфейса  
entityManager.saveRecord(budget2);  
entityManager.saveRecord(budget3);  
entityManager.saveRecord(budget4); // или реализацию интерфейса расширяющего интерфейс
```

```
}
```