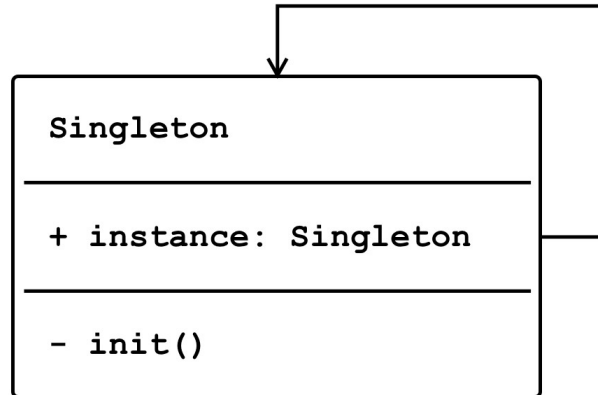


Фреймворк Spring. Конфигурация при помощи аннотаций.

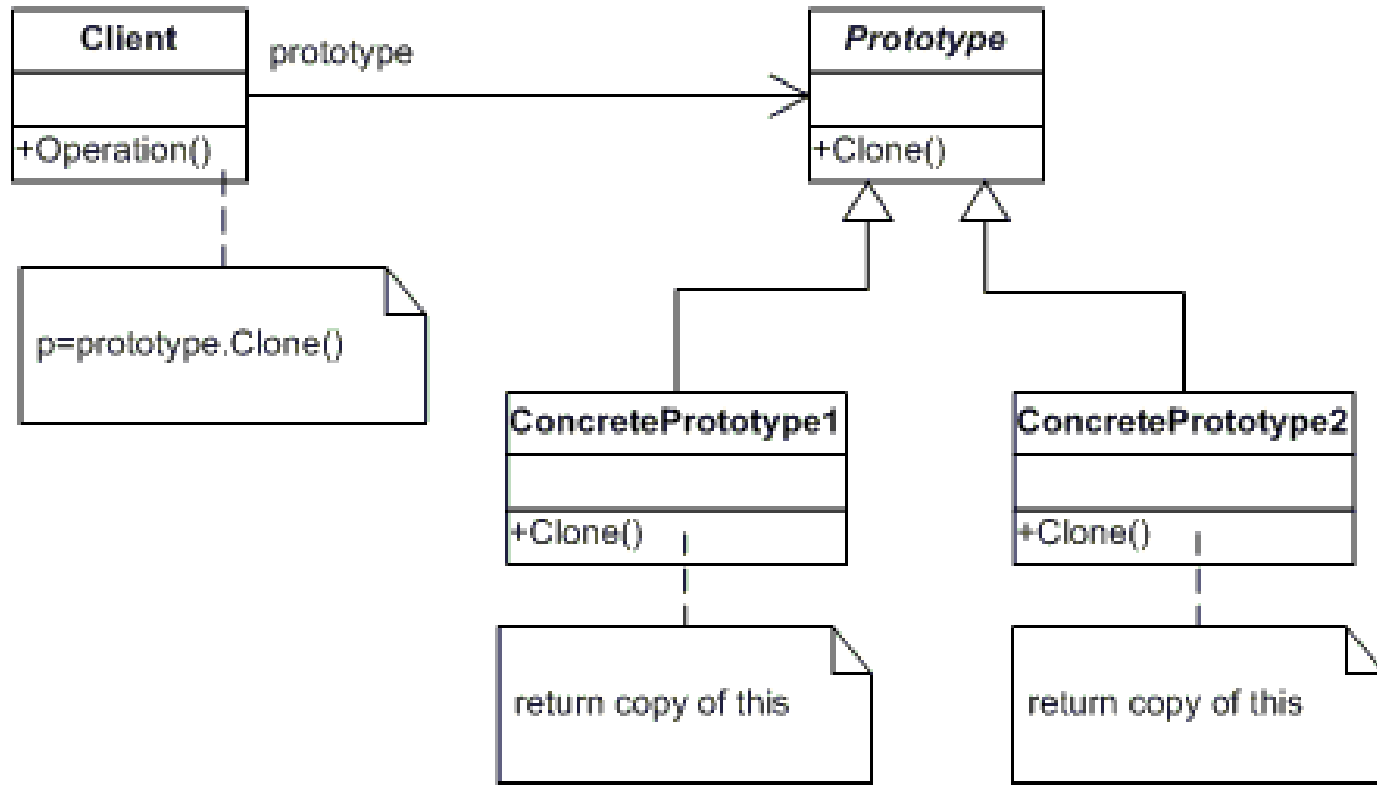
Составил: Черниговский А.С.
Старший преподаватель кафедры “Информатика”
ИКИТ СФУ



Паттерн Singleton (Одиночка)



Паттерн Prototype (Прототип)



Область действия КОМПОНЕНТОВ

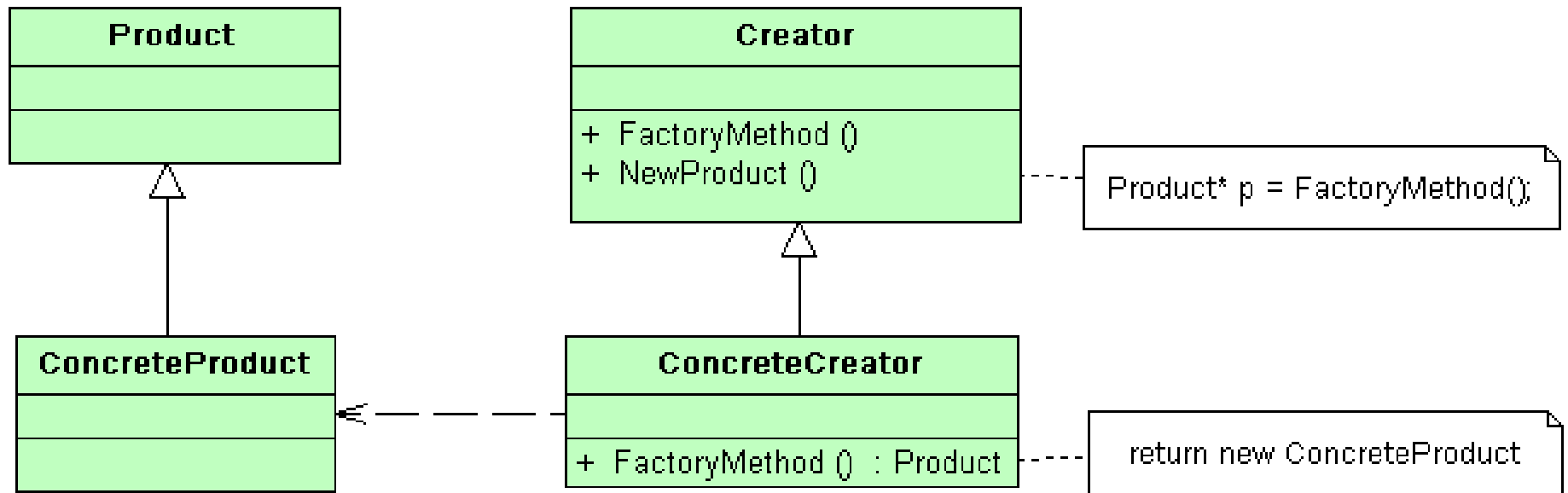
```
<bean id="engineBean"  
      class="ru.firstapp.GasEngine" scope="prototype">  
      <property name="volume" value="${gasEngine.volume}"/>  
</bean>
```

Инициализация и уничтожение компонентов

- Запуск Spring приложения → Запуск Spring контейнера → Создание объекта бина → Внедрение зависимостей в компоненту (бин) → Вызов init-метода → Работа приложения с готовыми бинами → Вызов destroy-метода → остановка Spring приложения
- ```
<bean id="lampBean"
 class="ru.firstapp.Lamp"
 init-method="turnOn"
 destroy-method="turnOff">

</bean>
```

# Фабричный метод



# Создание компонентов с помощью фабричных методов

- `private Lamp() {  
 }  
}`
- `public static Lamp getLamp(){  
 System.out.println("Фабричный метод. Хоба!");  
 return new Lamp();  
}`
- `<bean id="lampBean"  
 class="ru.firstapp.Lamp"  
 factory-method="getLamp">  
  
</bean>`

# Аннотации

- Аннотации – это пометки, с помощью которых программист указывает компилятору Java и средствам разработки, что делать с участками кода помимо исполнения программы. Аннотировать можно переменные, параметры, классы, пакеты. Можно писать свои аннотации или использовать стандартные – встроенные в Java.



# Возможности аннотаций

- автоматически создавать конфигурационные XML-файлы и дополнительный Java-код на основе исходного аннотированного кода;
- документировать приложения и базы данных параллельно с их разработкой;
- проектировать классы без применения маркерных интерфейсов;
- быстрее подключать зависимости к программным компонентам;
- выявлять ошибки, незаметные компилятору;
- решать другие задачи по усмотрению программиста.

# Связывание посредством аннотаций в Spring

- `<context:annotation-config />`

# Автоматическое определение компонентов

- `<context:component-scan>`

# Аннотирование компонентов для автоматического определения

- `@Component` – универсальная аннотация, указывающая, что класс является компонентом Spring;
- `@Controller` – указывает, что класс определяет контроллер Spring MVC;
- `@Repository` – указывает, что класс определяет репозиторий данных;
- `@Service` – указывает, что класс определяет службу; любая пользовательская аннотация, определенная с помощью аннотации `@Component`.

# Аннотация @Autowired

- @Component ("carBean")  
public class Car {  
 private GasEngine engine;  
  
 @Autowired  
 public void setEngine(GasEngine engine) {  
 this.engine = engine;  
 }  
 @Autowired  
 public Car(GasEngine engine) {  
 this.engine = engine;  
 }  
}

# Аннотация @Autowired

- @Autowired
- private GasEngine engine;

# Уточнение неоднозначных зависимостей.

## Аннотация @Qualifier

- ```
public Car(Engine engine) {  
    this.engine = engine;  
}
```
- ```
@Autowired
public Car(@Qualifier("gasEngine") Engine engine) {
 this.engine = engine;
}
```
- ```
@Autowired  
@Qualifier("gasEngine")  
private Engine engine;
```

Внедрение значений с помощью аннотаций

- `@Value("1.6")`
- `private double volume;`

Конфигурация Spring в программном коде на Java

- @Configuration

```
public class SpringConfig {  
    @Bean  
    public Engine gasEngine() {  
        return new GasEngine();  
    }  
}
```

Внедрение из файла свойств при помощи аннотаций

- `@PropertySource("classpath:gasEngine.properties")`

AnnotationConfigApplicationContext

- `AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext(SpringConfig.class);`

Спасибо за внимание!