

# Генерично програмиране в Java

Лектор:  
инж. Божидар Бацов

# Генерично програмиране

- Въведено в Java 5
- Счита се от мнозина за най-значимата промена в историята на Java
- Основната концепция е взаймствана от .NET, но имплементацията е коренно различна

# Нуждата от генерично програмиране

Илюстрирана с пример от живота...

# Класът ArrayList

- Клас от Collections API
- Динамично разширяем масив
- Предлага възможности за добавяне, изтриване на елементи, проверка за наличие на елемент и т.н.

Пример в NetBeans

# Проблеми на стандартната версия на класа

- Обектите, които съдържа са от тип `object`
- Необходими са преобразувания до типа, с който желаем да работим
- Няма ограничения за типа на обектите, които могат да бъдат поставени в `ArrayList`(няма `type safety`)

# Генеричен клас ArrayList<T>

- Декларацията на класа указва типа на елементите в него
- Не са необходими преобразувания
- В ArrayList<T> не могат да бъдат поставени несъвместими с T типове
- T – типова променлива(клас), например String, Integer, Person и т.н.

Пример в NetBeans

# Аспекти на генеричното програмиране

- Използване на съществуващи генерични класове/методи
- Анализирание на проблеми в/със съществуващи генерични класове/методи
- Създаване на собствени генерични класове/методи

# Дефиниране на генеричен клас

- Генеричния клас има в декларацията си една или повече типови променливи
- Типовите променливи се поставят в `<>`, след името на класа
- При инстанциране на класа ние замествахме типовите променливи с конкретни типове (String, Double, etc)

## Дефиниране на прост генеричен клас

```
class GenericClass<T> {  
    private T someField;  
    public T someMethod() {  
    }  
}  
  
GenericClass instance = new  
GenericClass<String>();
```

Пример в NetBeans

# Дефиниране на генеричен метод

- Може да дефинирате генерични методи в обикновени(негенерични) класове
- В декларацията на метод трябва трябва да добавите в <> генеричните типове преди връщата стойност от метода
- Когато извиквате генеричен метод може да поставите типовете промениливи преди името на метода(но това обикновено не се налага)

# Дефиниране на генеричен МЕТОД

```
class SomeClass {  
    public static <T> T someMethod(T param) {  
    }  
}  
  
SomeClass.<String>someMethod("ala");  
SomeClass.someMethod("bala");
```

Пример в NetBeans

# Граници(bounds) за типови променливи

- Понякога се налага да поставите ограничения на това кои типове могат да бъдат използвани като стойност на типови променливи в генеричен клас – например може да искате само типове, които имплементират Comparable
- T extends Comparable

Пример в NetBeans

# Генеричен код и Java виртуалната машина

- В JVM няма генерични класове – само обикновени
- Generics в Java са имплементирани като функционалност на компилатора, а не на самата виртуална машина
- Всеки генеричен тип има кореспондиращ “суров” клас (raw class) – в него типовете променливи са изтрети и са заменени от техния ограничащ тип

# Генеричен код и JVM

- Ако няма ограничаващ тип – замяната е с Object
- Когато имаме повече от един тип – замяната с първия, а за другите компилатора добавя преобразувания в кода, където е необходимо

```
class SomeClass<T extends Serializable &  
Comparable>
```

# Работа със стар код(преди Java 5)

- Когато работите в Java 5 или Java 6 със стар код, който използва “сурови” типове ще се сблъсквате с предупреждения за потенциални проблеми от компилатора
- `@SuppressWarnings(“unchecked”)` - тази анотация потиска тези предупреждения

# Ограничения

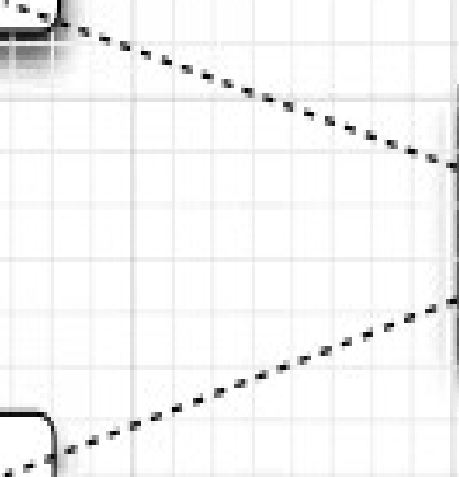
- Типов параметър не може да бъде примитивен тип
- Губи се информацията за типа по време за изпълнение – `ArrayList<String>.class`  
`=== ArrayList<Double>.class`  
`ArrayList.class`
- Не могат да се създават масиви от параметризирани(генерични) типове

# Ограничения

- Не може да инстанцирате типовите променливи
- Типовите променливи не са позволени в статичен контекст
- Внимавайте за конфликти на имена в резултат на типовото изтриване по време на изпълнение на програмите

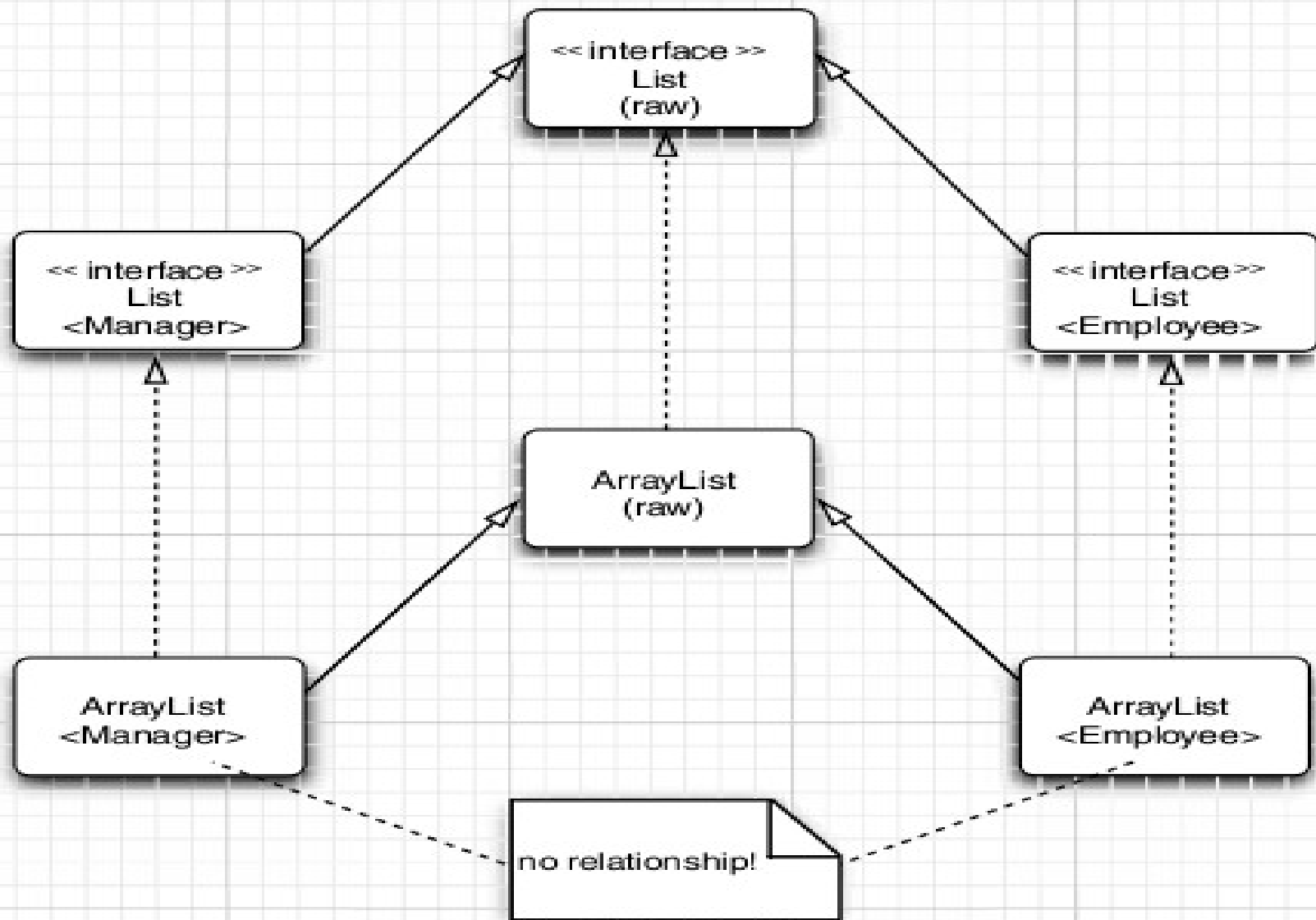
# Правила за наследяване

- Генеричните класове с типови параметри, които са свързани с наследяване (например `Employee` и `Manager`) не са свързани по между си
- `Pair<Employee>` и `Pair<Manager>` нямат никаква връзка помежду си, доколкото наследяването е свързано



no relationship!

A note box with a folded top-right corner, containing the text "no relationship!".



# Заключение

- Генеричните типове и методи ни дават допълнително ниво на сигурност и гъвкавост
- Използвайте винаги генеричните версии на класовете, ако съществуват такива
- Не забравяйте, че в Java generics е реализиран на ниво компилатор – виртуалната машина, не знае почти нищо за генеричните типове

Въпроси?

Благодаря Ви за вниманието :-)