# Обобщенное программирование

- обработка не требует информации о типе данных → воспринимаем как контейнеры
- В классе можно использовать универсальные поля Object
- Неудобно, т.к. нужно испольщовать приведение типов. Можем получить RE
- Дженерики
- В определении класса не указываем типы, при создании указываем
- Diamond operator <> позволяет не дублировать типы при вызове конструктора
- Примитивы нельзя использовать с дженериками
- Параметры можно задавать не только для классов, но и для отдельных методов
- Ограничения (bounds)
- Ограничение типов параметров:
  - <T extends Number>
  - <T extends Iterator & Closeable>
- <?>
- Pair<String, Integer> не является наследником Pair<Object, Object> → можно применить wildcard-тип void print(Pair<?, ?> pair)
- Ограничения не только сверху но и снизу:
  - <? extends Number>
  - <? super Integer>
- Параметризованный класс без спецификации парамтеров называется raw-классом.

  Raw и параметризованные типы можно неявно приводить между собой.
- Heap pollution: задекларированные значения типа параметра не соответствует его значению в рантайме.
- Ограничения (limitations):
- Что нельзя делать с параметрами типов и методов:
  - Использовать в качестве типов статических полей классов
  - Создавать экземпляры (Но есть Class<T>.newInstance())
  - Создавать массивы
  - Применять оператор преобразования типов ()
  - Применять оператор instanceof

# hashCode & equals

- По умолчанию Object.equals сравнивает ссылки → переопределение equals в классе
- Класс Object определяет метод, позволяющий вычислить хеш объекта  в виде значения типа int.
- Если x.equals(y), то x.HashCode() == y.hashCode() → Переопределение hashCode
- Always override hashCode when you override equals

# Коллекции

- java.util.Collection
- Три кита:
  - Список: java.util.List
  - Словарь: java.util.Map
  - Множество: java.util.Set
- Лист поддерживает вставку элемента по индексу
- *ArrayList* хранит элементы в массиве, который может увеличиваться при необходимости. *LinkedList* основан на двусвязном списке.
- *HashMap* разбивает ключи на группы по хешам объектов и формирует на их основе списки адресуемых значений. Проблема: коллизии хешей. *TreeMap* основана на красно-черных деревьях. То есть ключи хранятся в упорядоченном виде
- *HashSet* и  *TreeSet*