ジェネリクスを使おう！

凪瀬悠輝（なぎせゆうき）

簡単なジェネリクス

```
List<String> list =
    new ArrayList<String>();
```

コレクションとしてのジェネリクス

- List<T>とか
- Map<K, V>とか
- データの集合を表すクラスで、データの型をジェネリクスで表現する使い方
- 一番分かりやすい
- なじみ深い
- 序の口

ジェネリックなクラスを作る

Class宣言の横に型変数を宣言する

変数宣言時に型変数を使えるようになる

```
public class Hoge<T> {
    public T value;
}
```

型変数はインスタンス化の際に型をバインド

```
Hoge<String> hoge = new Hoge<String>();
hoge.value = “hoge”;
String str = hoge.value;
```

ラッパークラスとしてのジェネリクス

例えばResult<T>という型を考える

```
/** 処理結果のラッパークラス */
public class Result<T> {
    public T value;
    public Status status;
    public String errorMessage;
}
```

ジェネリックなクラスの継承

継承の際に型変数に型をバインドする

```
public class StringList
    extends ArrayList<String> { … }
```

型変数に型変数をバインドするケース

```
public class HogeList<A, B, C>
    extends ArrayList<B> { … }
```

型変数の境界

型変数は宣言する時に境界を設定できる

```
public class Hoge<T extends B> { … }
```

C extends B、B extends A のとき

```
new Hoge<A>(); // NG
new Hoge<B>(); // OK
new Hoge<C>(); // OK
```

型変数の境界 その2

ある型階層より上という指定の仕方

```
public class Piyo<T super B> { … }
```

C extends B、B extends A のとき

```
new Piyo<A>(); // OK
new Piyo<B>(); // OK
new Piyo<C>(); // NG
```

型変数の境界　その3

public class Foo<T extends C & I> { … }

クラス C を継承して、且つ、
インターフェース I を継承する型のみを
型変数Tにバインドすることができる。

ワイルドカード

今度は型変数にバインドする型引数の話
型引数にはワイルドカードが使用できる

- List<?> list;
- List<? extends B> extendsList;
- List<? super B> superList;

代入互換性

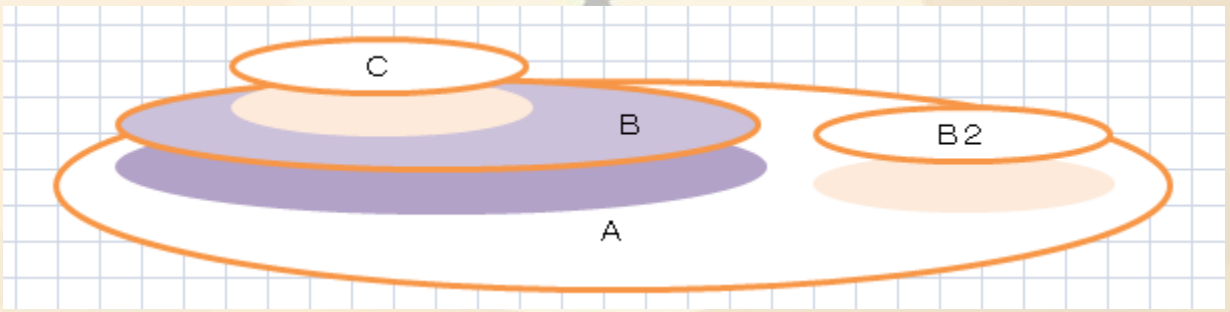
集合の代入になるのでOOP的代入と異なる

List<B> listB = new ArrayList<C>(); // NG

なぜかというと、listB.add(new B());

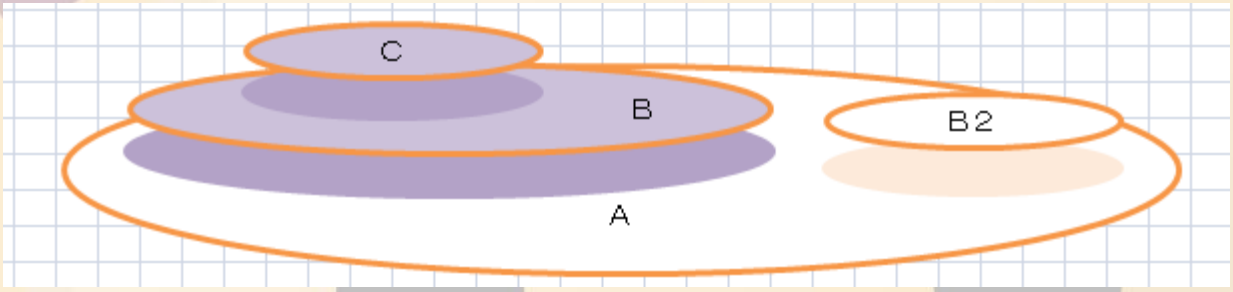
とするとArrayList<C>にB型が

add()されてしまうから。

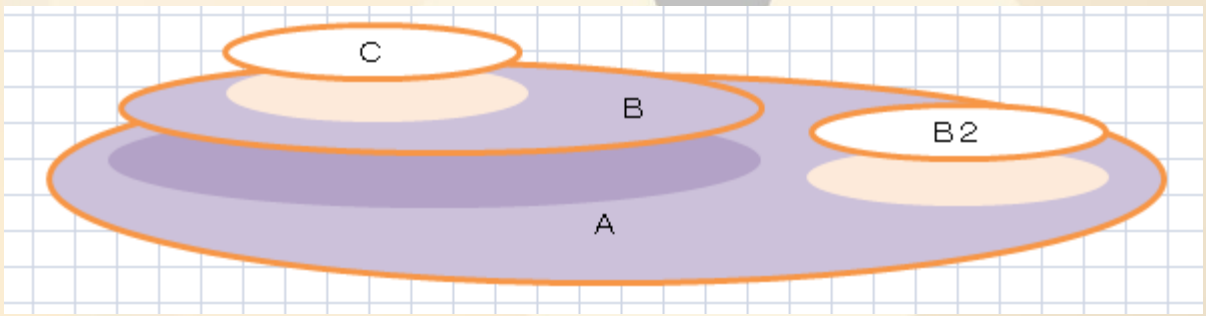


代入互換性　その2

List<? extends B>の範囲

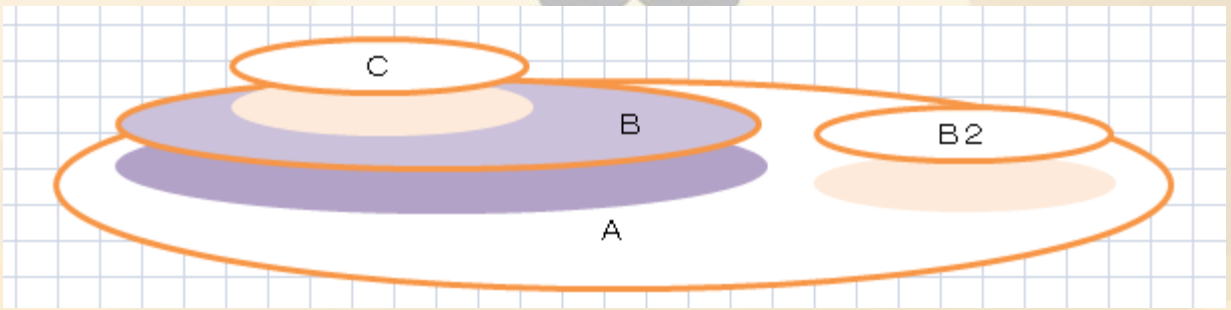


List<? super B>の範囲



代入互換性　その3

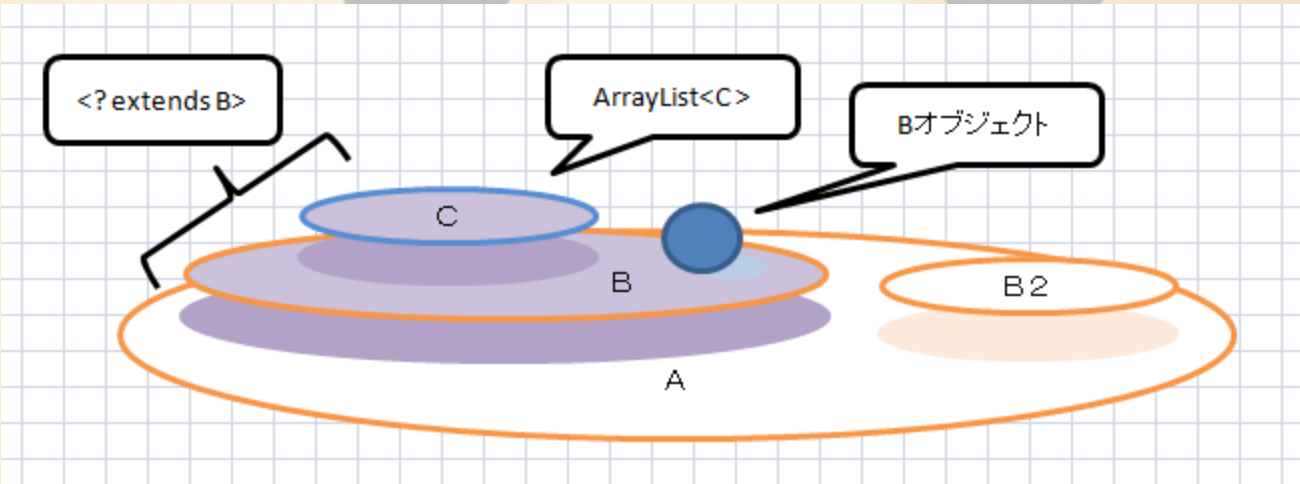
- List<? extends B>型に
  - List<B>が代入可能
  - List<C>が代入可能
- List<? super B>型に
  - List<B>が代入可能
  - List<A>が代入可能



ワイルドカード使用時の引数の制限

List<? extends B>にはadd()できない

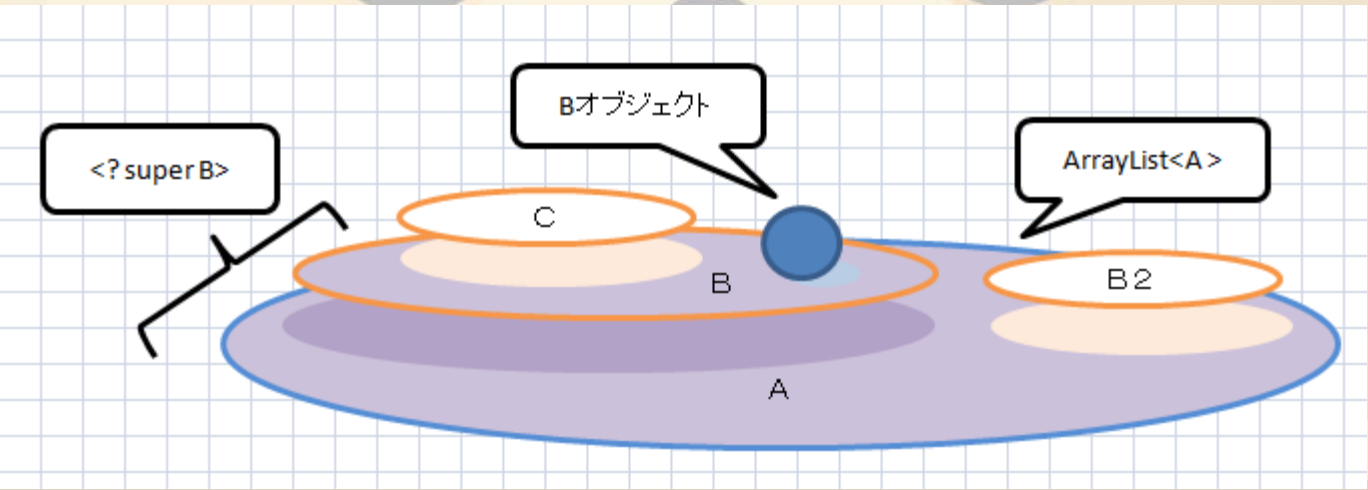
- ArrayList<C>型を代入
- B型をadd()とする
- List<C>にB型がadd()される　←矛盾



ワイルドカード使用時の引数の制限 その2

List<? super B>にはB型をadd()できる

List<? super B>に代入可能なList<A>は
B型をadd()しても大丈夫



ワイルドカード使用時の戻り値の制限

List<? extends B>からはB型をget()できる

- B b = list.get(0);

List<? super B>からは
Object型でしかget()できない

- Object o = list.get(0);

List<A>とかが代入されてるかもしれないから