

How can you use both mutable and immutable Sets in the same file?

```
import scala.collection.mutable  
mutable.Set
```

Or rename one of them.

Describe the `x += y` compiler magic.

Why is it provided?

What does it apply to?

It will use += operator if possible. Otherwise it will try to expand to

$$x = x + y$$

It applies to **any** assignment method to make it easy to switch from mutable to immutable collections and vice versa.

Create a collection from another of a different type.

Create an empty one and then add (++) to the existing one.

```
val treeSet = TreeSet[String]() ++ list
val mutaSet = mutable.Set.empty ++ treeSet
val immu = Map.empty ++ muta
```

What do Tuples save you?

The tedium of defining simplistic data-heavy classes.



Name the default mutable and immutable  
`Sets/Maps` provided by factory methods.

`HashSet` and `HashMap` for all mutable sets, maps, and large immutable ones.

Small (0-4 elements) immutable `Sets` and `Maps` get specialized implementations like `Set4`.

Describe the behavior of

```
val first, second = someTuple2
```

It will multiply assign the `Tuple2` to both first and second, since there are no parens.

Describe four ways to make threadsafe sets and maps.

- Mix in `SynchronizedSet` **or** `SynchronizedMap`.
- **Use** `java.util.concurrent`.
- Use actors and unsynchronized collections.
- Or, use immutable ones.

What can newer imports do to older ones?

They can hide clashing simple names of the older ones

```
Map()    //immutable
import  scala.commection.mutable.Map
Map()    //mutable
import  scala.collection.immutable.Map
Map()    //immutable
```



What is an important difference between  
Scala and Java string?

Scala's implicitly converts to `RichString` which is `Seq[Char]`, hence you can iterate over it or use `Seq` methods on it.

Give a nicer way to widen the static type of a new collection.

```
val stuff = Set[Any] (42)
```

**Nicer than**

```
val stuff: Set[Any] = Set(42)
```

Sorted sets and maps are implemented with  
which classes?

They are always ...

Which structure do they use?

TreeSet **and** TreeMap.

... immutable.

They use red-black trees on items (or keys).

How does `Predef` choose the default interpretation of `Set`?

- It defines the type alias of `Set[T]`.
- It makes a `val Set` the singleton

`scala.collection.immutable.Set.`

```
typeSet[T] = scala.collection.immutable.Set[T]  
val Set = scala.collection.immutable.Set
```