AND450

Building a Java Bindings Library

Download class materials from university.xamarin.com

Microsoft

Xamarin University

Objectives

1. Determine which Java libraries can be bound
2. Bind a .JAR
3. Bind an .AAR
4. Handle a private-use reference .JAR
5. Handle a public-use reference .JAR
6. Examine how Java-language constructs are mapped to C#

Determine which Java libraries can be bound

Tasks

1. Define the term *Bindings Library*
2. Describe which libraries can be bound

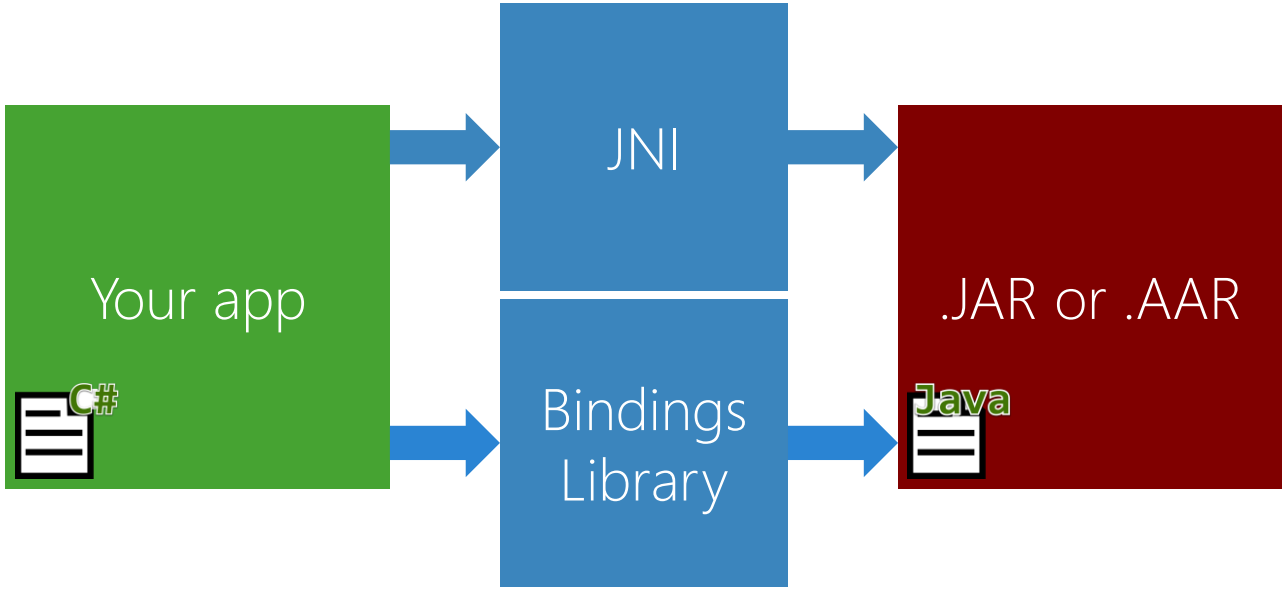
Motivation

- The Android community has many Java libraries you might want to use in your app

ArcGIS	PayPal	TritonPlayer	…
Mapping	Finance	Music	

Java integration options

- You can use JNI or a Bindings Library to incorporate Java libraries into your Xamarin.Android app



What is JNI?

- The *Java Native Interface* (JNI) enables interop between Java and other languages

```java
package com.xamarin.mycode;

public class MyClass
{
  public String myMethod(int i) { ... }
}
```

JNI is verbose: this constructs an object and calls a method

```csharp
IntPtr type        = JNIEnv.FindClass("com/xamarin/mycode/MyClass");
IntPtr constructor = JNIEnv.GetMethodID(type, "<init>", "()V");
IntPtr instance    = JNIEnv.NewObject(type, constructor);
IntPtr method      = JNIEnv.GetMethodID(type, "myMethod", "(I)Ljava/lang/String;");
JValue argument    = new JValue(17);
IntPtr result      = JNIEnv.CallObjectMethod(instance, method, argument);
```

What is a Bindings Library?

- A *Bindings Library* is an assembly containing Managed Callable Wrappers for Java types

```
package com.xamarin.mycode;

public class MyClass
{
  public String myMethod(int i) { ... }
}
```

Java

The Bindings Library defines this class and this method

```
var instance = new MyClass();

string result = instance.MyMethod(17);
```

C#

Bindings implementation

- Bindings libraries use JNI internally to interoperate with Java

Your code →

```
string result = instance.MyMethod(17);
```

Binding code →

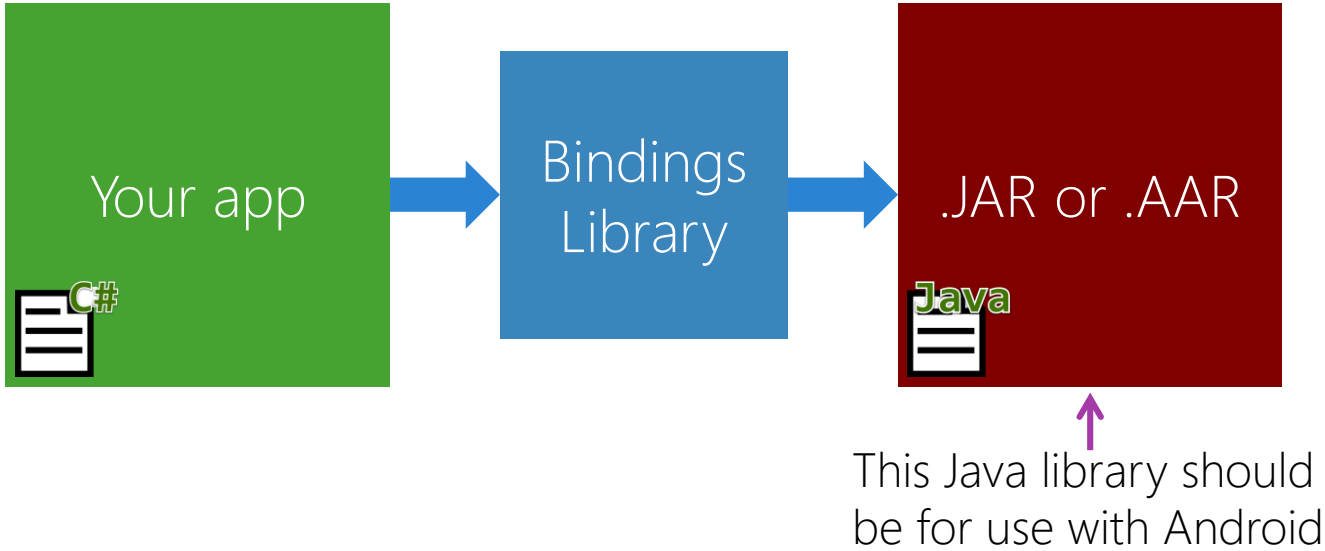
```
IntPtr m = JNIEnv.GetMethodID(...);
JValue a = new Jvalue(...);
IntPtr r = JNIEnv.CallObjectMethod(...);
```

.JAR
or
.AAR

Bindings Library

Android libraries only

- Bindings are for Java libraries that were built for Android



Binding non-Android Java libraries may work but is not an officially-supported scenario

Flash Quiz

Xamarin University

Flash Quiz

① You can create Bindings Libraries for .JAR files and .AAR files

a) True

b) False

Flash Quiz

① You can create Bindings Libraries for .JAR files and .AAR files

a) True

b) False

Flash Quiz

② You can create a Bindings Library for any Java library

a) True

b) False

Flash Quiz

② You can create a Bindings Library for any Java library

a) True

b) False

Summary

1. Define the term *Bindings Library*
2. Describe which libraries can be bound

Bind a .JAR

Tasks

1. Create a Bindings Library for a .JAR
2. Use the Bindings Library types

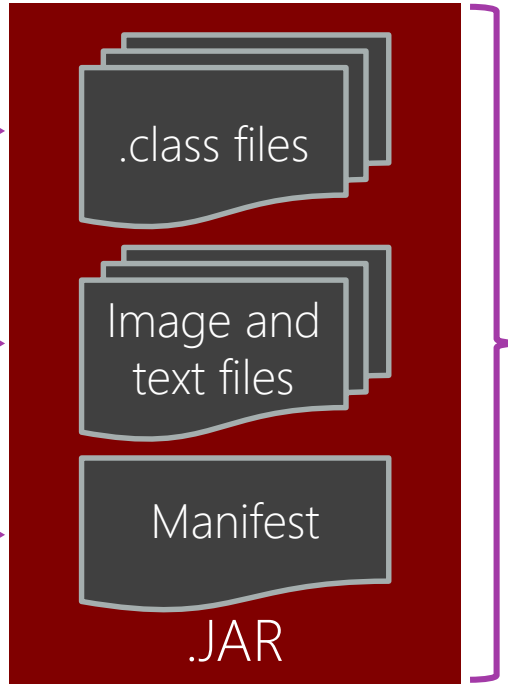
What is a .JAR file?

- A *Java Archive (.JAR)* is the unit-of-deployment for Java apps and libraries

Compiled Java code

Java Resources (Note: not Android resources)

Java manifest containing metadata like the identity of the main class (Note: not an Android manifest)



Uses .ZIP file format

What is a Managed Callable Wrapper?

- A *Managed Callable Wrapper* is a C# type that wraps a Java type, operations on the wrapper class are forwarded to the original Java type

```
public class Contact
{ ...
  public String getPhoneNumber(String category)
  {
    ...
  }
}
```

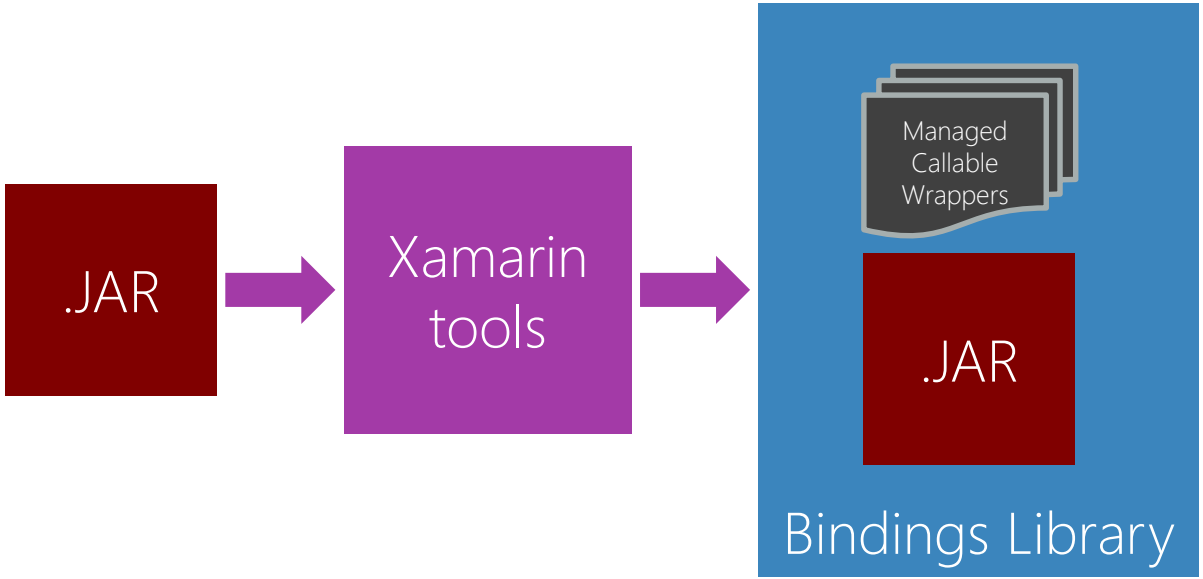
Original Java type

```
public class Contact : Java.Lang.Object
{ ...
  public virtual string GetPhoneNumber(string p0)
  {
    ...
  }
}
```

C# wrapper class

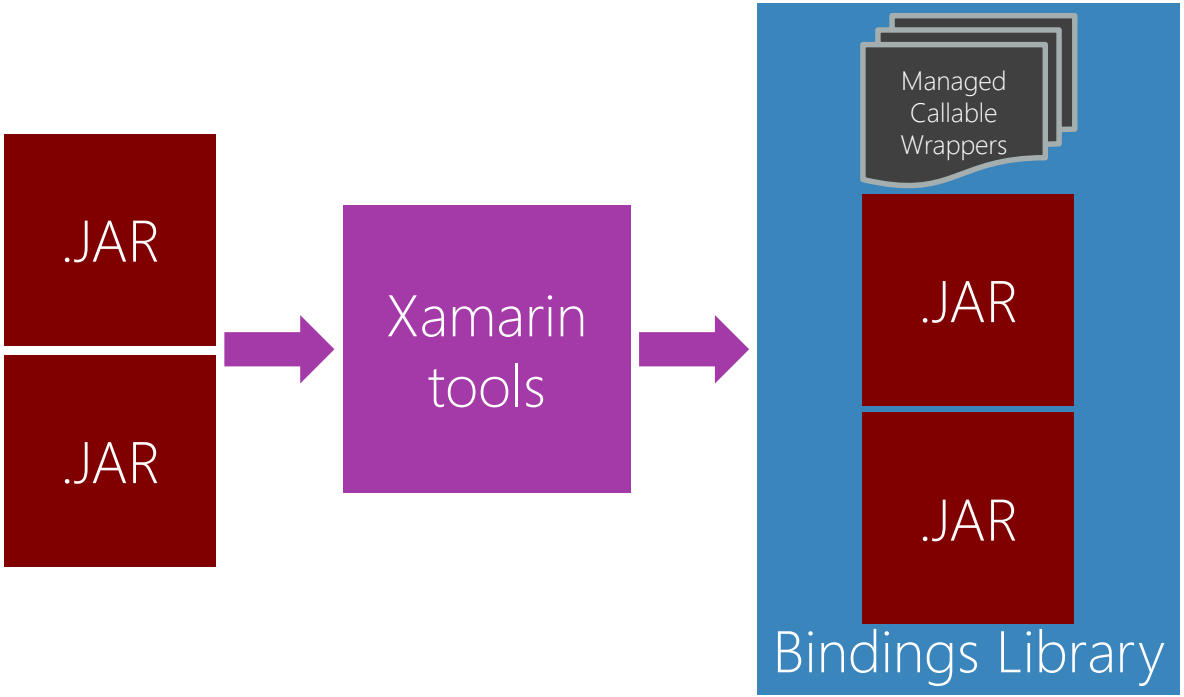
Automatic generation

- Xamarin tools generate a Bindings Library from an input .JAR



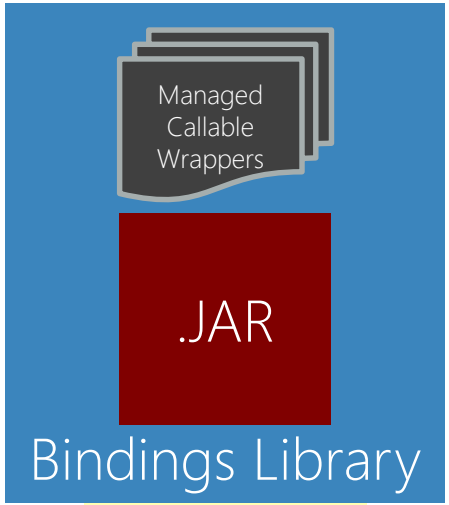
Multiple input files

- A single Bindings Library can contain bindings for many .JAR files



Build Action

- **Build Action** determines if the .JAR will be embedded in the Bindings Library



`EmbeddedJar`

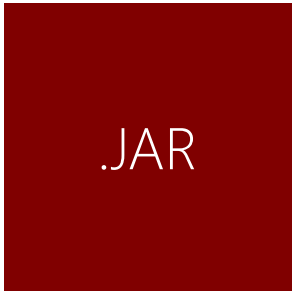
`InputJar`

Embedding is preferred in practice because deployment is easy: the necessary .JARs will be compiled to Dalvik bytecodes, packaged in the .APK, and available to the app at runtime.

How to create a .JAR Bindings Library

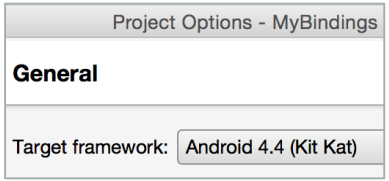
- There are several steps to create a Bindings Library

1. Create a new project



2. Add your .JAR(s)

3. Set the Build Action



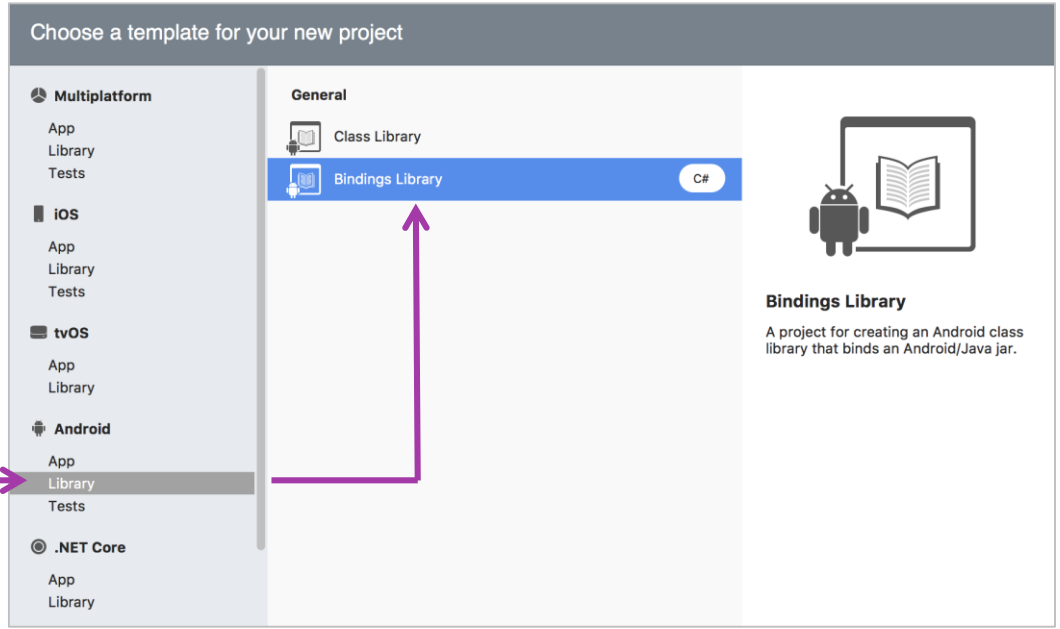
4. Choose the target framework

5. Build

How to create a .JAR Bindings Library [1]

- Create a Bindings Library project in Visual Studio

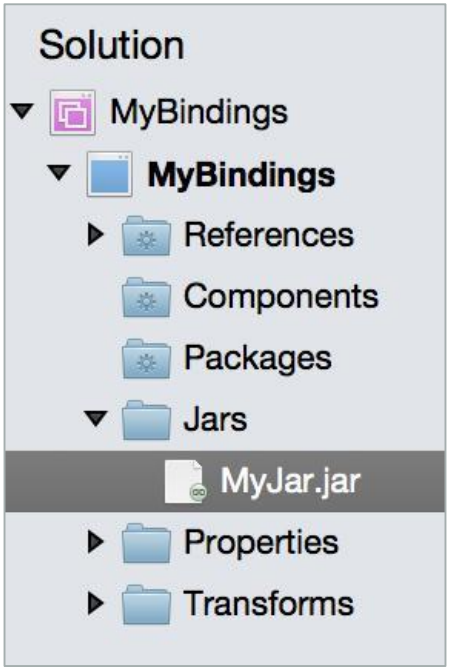
The template is in the Android Library category



How to create a .JAR Bindings Library [2]

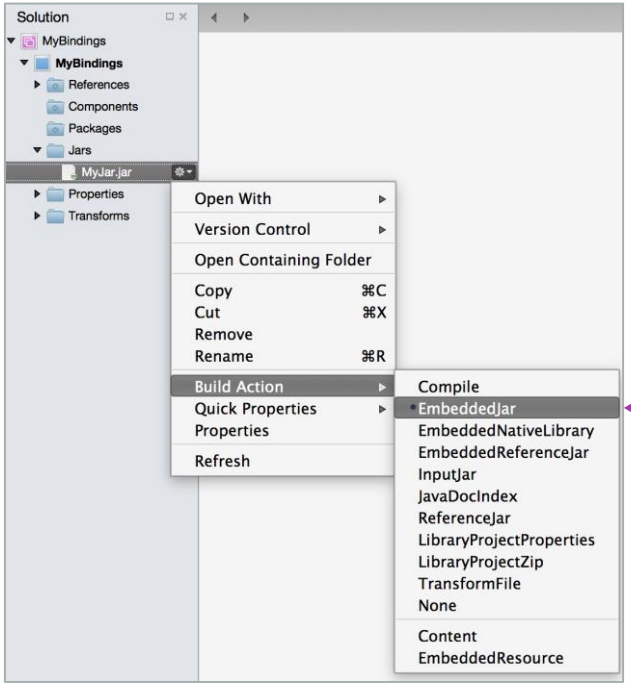
- Add your .JAR files to the project

Place your .JAR files in the "Jars" folder



How to create a .JAR Bindings Library [3]

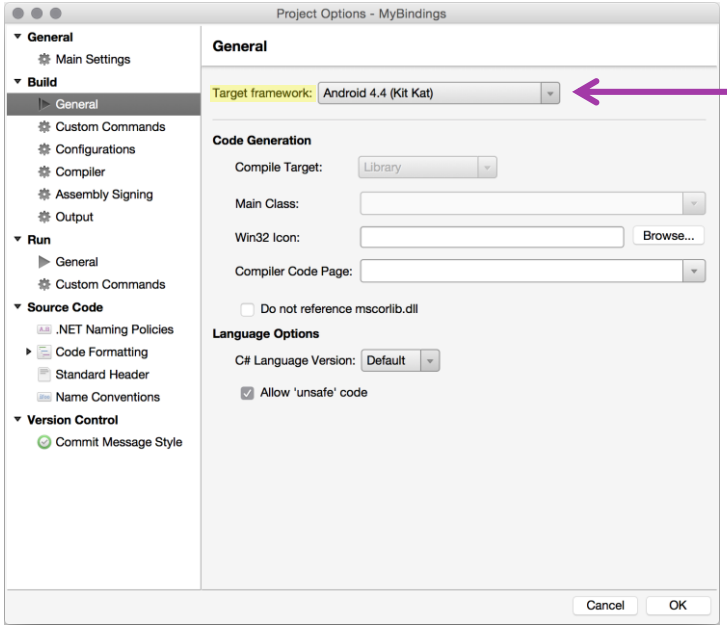
- Set the Build Action for each .JAR file



Set the Build Action to **`EmbeddedJar`** or **`InputJar`**

How to create a .JAR Bindings Library [4]

- Set the Target framework version for your Bindings Library



E.g. usable with 4.4 and newer apps

How to create a .JAR Bindings Library [5]

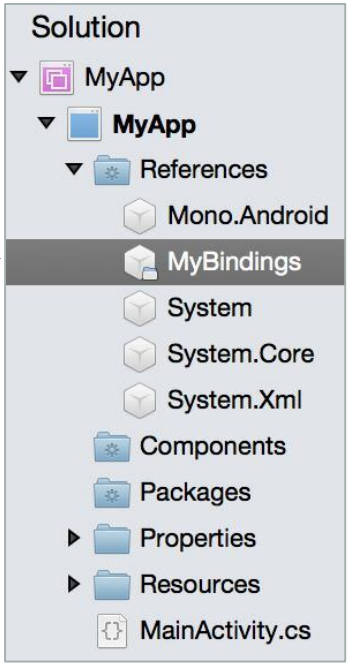
- Build the project, a Bindings Library project generates an assembly

Output is a .DLL containing the Managed Callable Wrappers

How to use a .JAR Bindings Library

- Bindings Libraries are .DLLs, you reference them in your Android project

1. Add a reference




```
var c = new Com.Mycompany.Mynamespace.MyClass();
c.MyMethod();
```

2. Use the types

Individual Exercise

Bind a .JAR

Xamarin University

Summary

1. Create a Bindings Library for a .JAR
2. Use the Bindings Library types

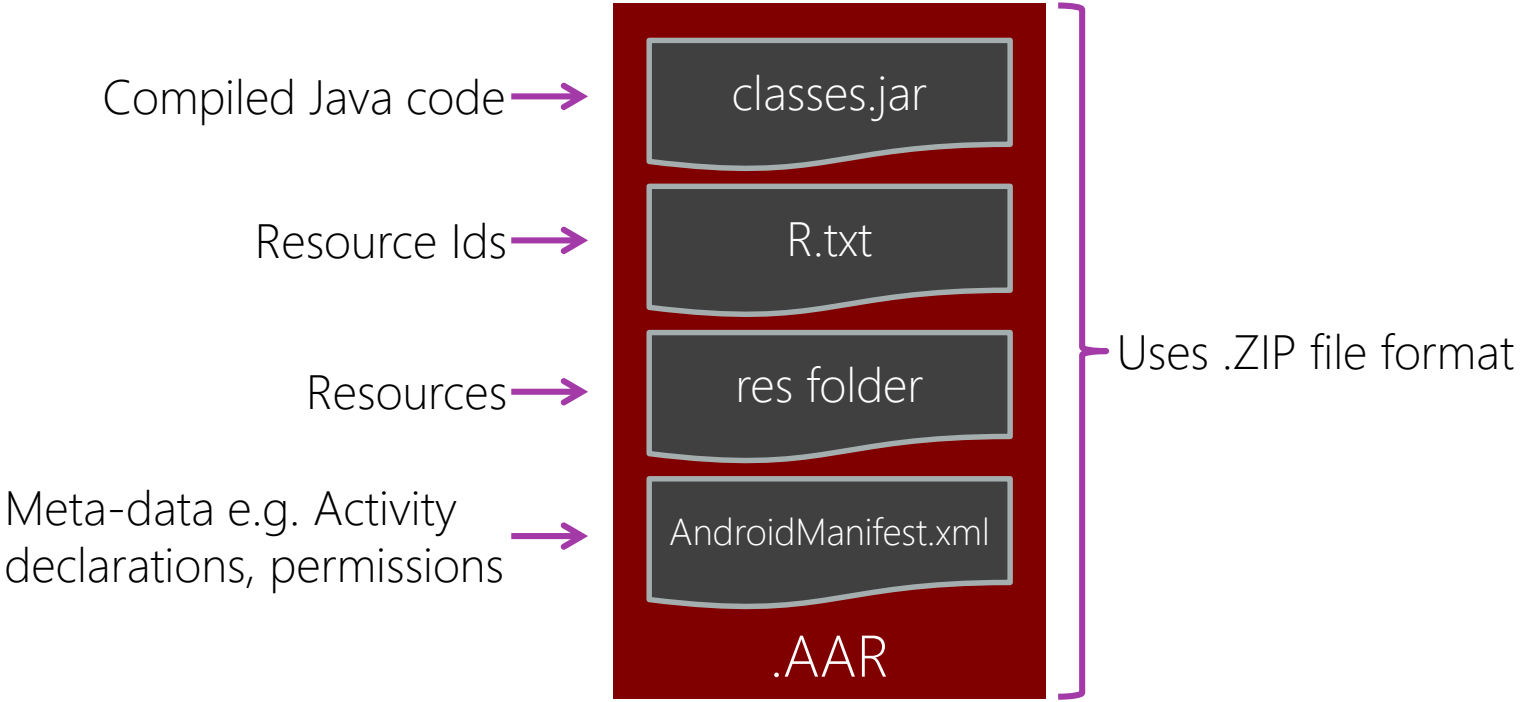
Bind an .AAR

Tasks

1. Create a Bindings Library for an .AAR
2. Use the Bindings Library types
3. Use the Bindings Library resources

What is an .AAR file?

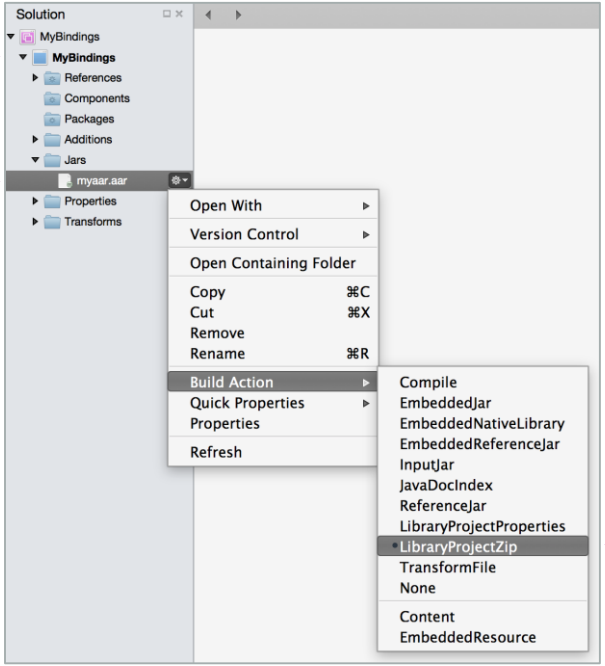
- An *Android Archive (.AAR)* file is the file format for Android libraries



How to create an .AAR Bindings Library

- Binding an .AAR file is the same as a .JAR except for the Build Action

1. Create an **Android Java Bindings Library** project
2. Add your .AAR files to the "Jars" folder



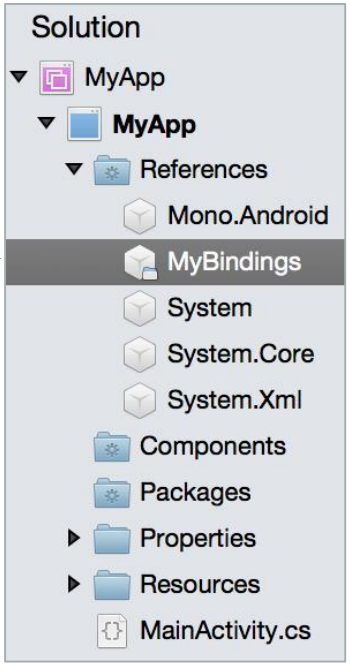
3. Set the Build Action to `LibraryProjectZip`

The .AAR will be embedded in the .DLL, this is the only option for .AAR files

How to access .AAR types

- Access to the types in an .AAR Bindings Libraries is the same as for .JARs

1. Add a reference

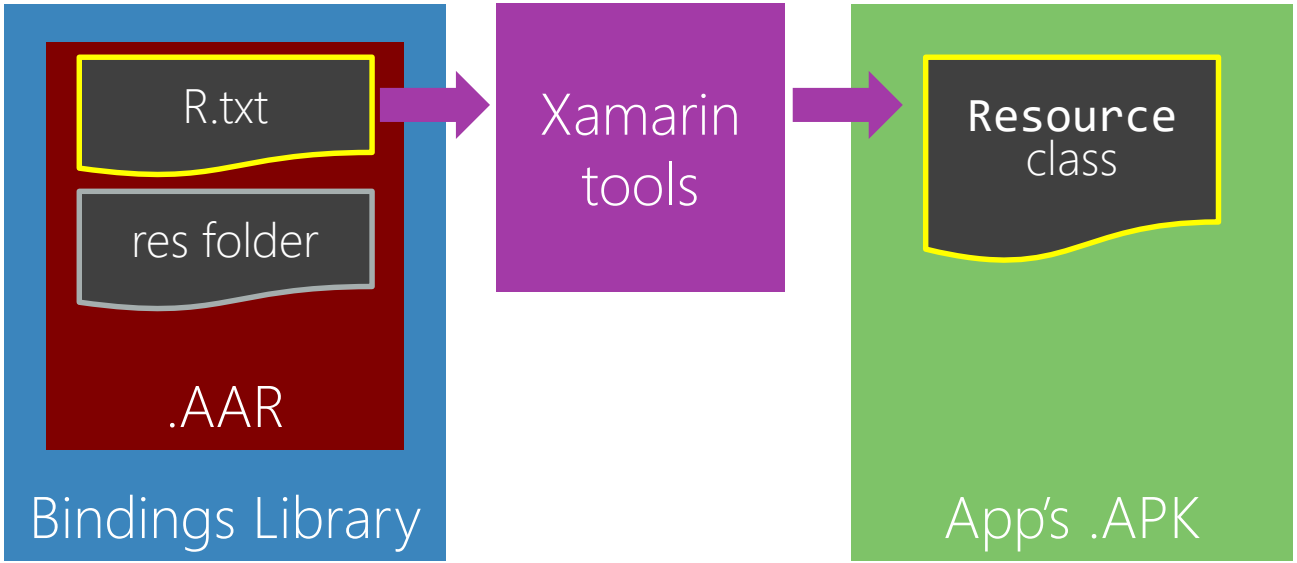



```
var c = new Com.Mycompany.Mynamespace.MyClass();
c.MyMethod();
```

2. Use the types

How to access .AAR resources

- You can access the resources in a bound .AAR file, the Xamarin tooling merges the **R** data from the .AAR into your App's **`Resource`** class



How to access an .AAR image

- Use the **`Resource.Drawable`** name for images packaged in the .AAR

Java

```
public final class R
{ ...
  public static final class drawable { public static int logo=0x7f020000; }
}
```

Java values are merged into the C# Resource class

C#

```
public partial class Resource
{ ...
  public partial class Drawable { public const int logo=2130837505; }
}
```

```
<ImageView android:src="@drawable/logo" ... />
```

How to access an .AAR layout

- Use the **`Resource.Layout`** name for layouts packaged in the .AAR

Java

```
public final class R
{ ...
  public static final class layout { public static int row_layout=0x7f030001; }
}
```

Java values are merged into the C# Resource class

C#

```
public partial class Resource
{ ...
  public partial class Layout { public const int row_layout=2130903042; }
}
```

```
var a = new ArrayAdapter<string>(this, Resource.Layout.row_layout, ...);
```

How to access an .AAR string

- Use the **`Resource.String`** name for strings packaged in the .AAR

Java

```
public final class R
{ ...
  public static final class string { public static int greeting=0x7f040000; }
}
```

Java values are merged into the C# Resource class

C#

```
public partial class Resource
{ ...
  public partial class String { public const int greeting=2130968576; }
 }
```

```
<TextView android:text="@string/greeting" ... />
```

Individual Exercise

Bind an .AAR

Xamarin University

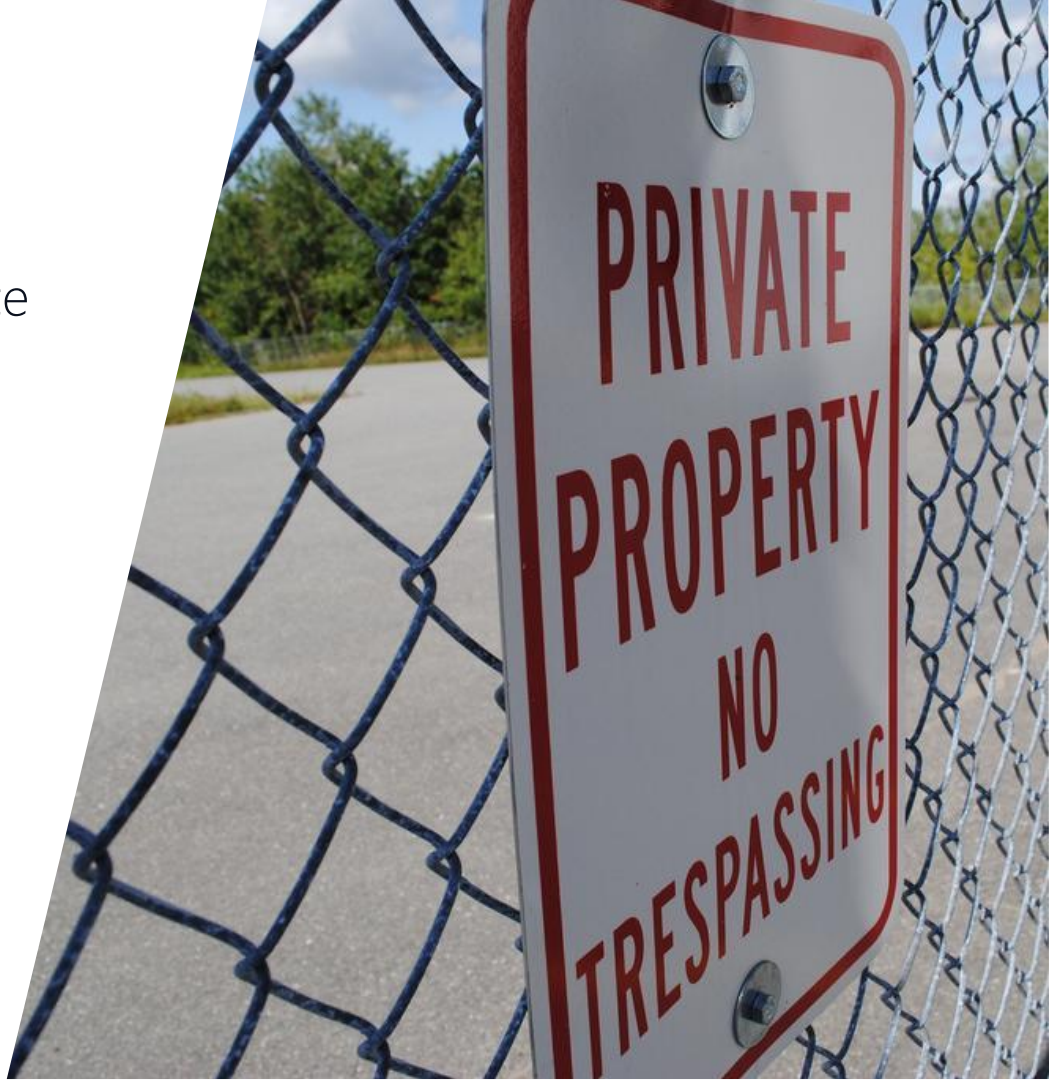
Summary

1. Create a Bindings Library for an .AAR
2. Use the Bindings Library types
3. Use the Bindings Library resources

Handle a private-use reference .JAR

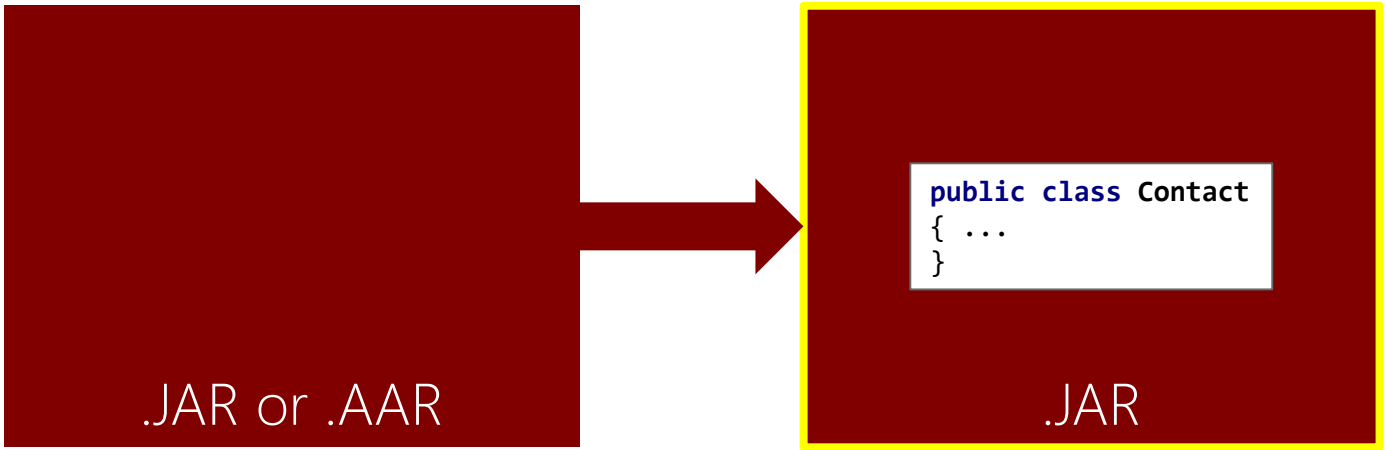
Tasks

1. Include a private-use reference .JAR in your Bindings Library



What is a reference .JAR?

- A *reference .JAR* is a .JAR file used by one of your bound .JAR or .AAR files



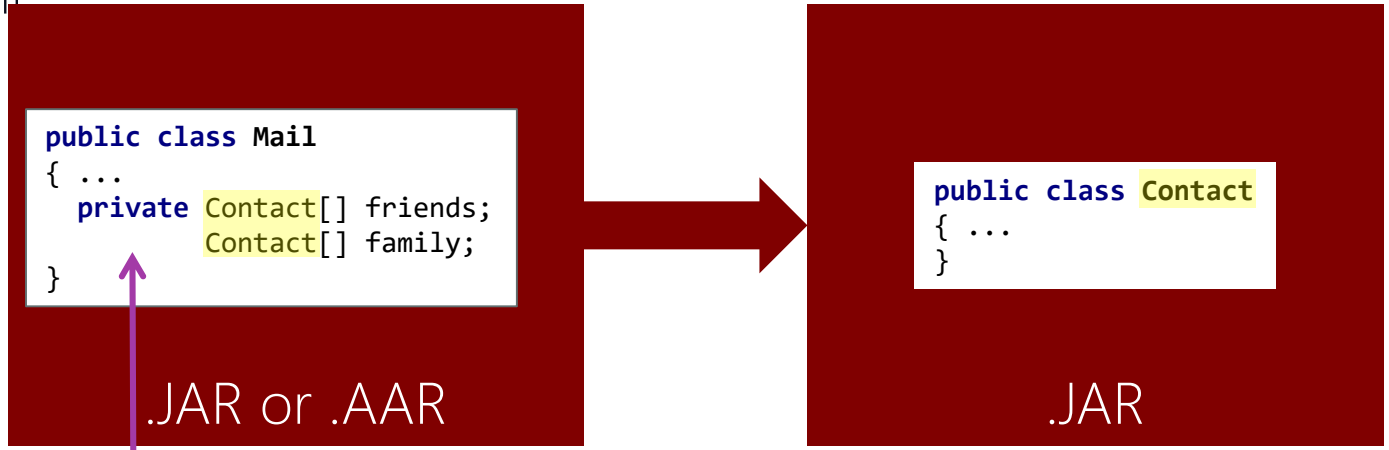
This is the library you need to create bindings for…

…however, it uses types defined here, this is the **reference .JAR**

Only .JAR files can be references, .AAR files cannot

What is a private-use reference .JAR?

- A reference .JAR is *private-use* if it only uses the **types from the reference .JAR** behind-the-scenes and does not expose them to the client

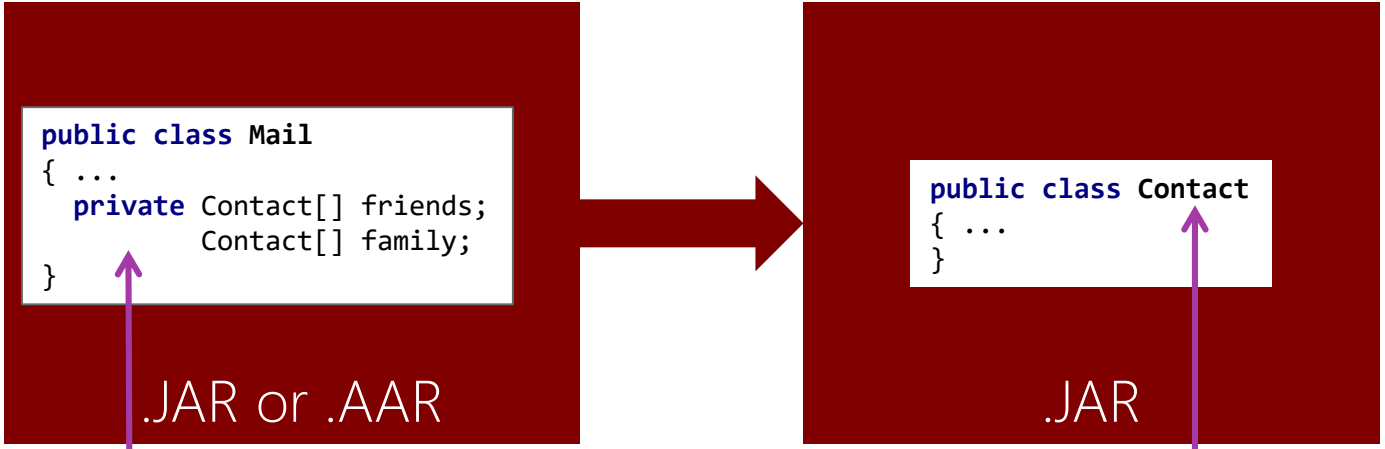


You use the types only as private fields, package fields, local variables, etc.

Note: the term *private-use* is used in this course, but is not an official phrase

No bindings needed

- You do not need to create Managed Callable Wrappers for the types in a private-use reference .JAR, the types are not exposed to the client

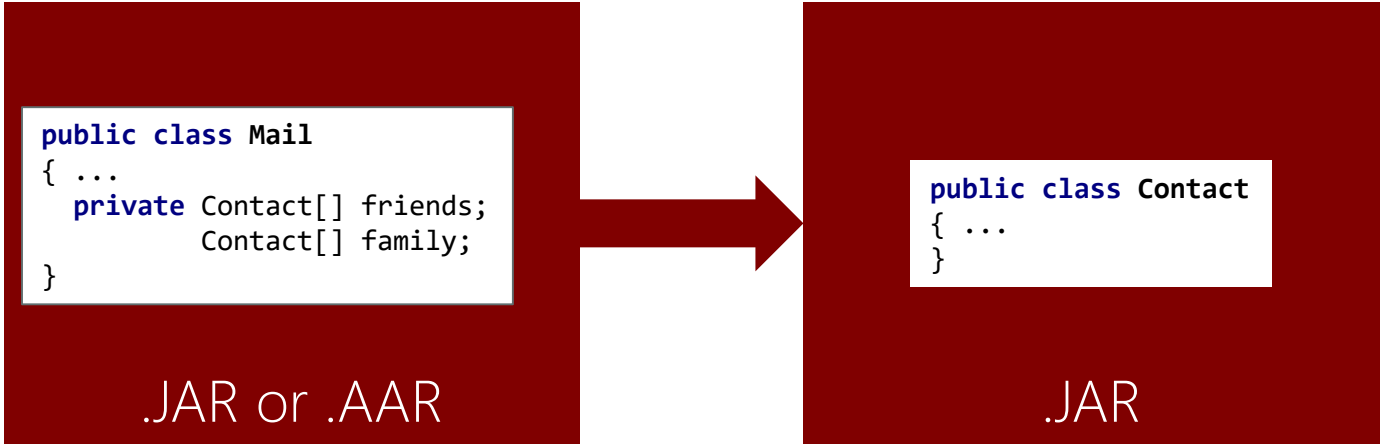


If this is private-use only…

… then no wrappers are needed for this class

Compile-time requirements

- For private-use reference .JARs, there are no compile-time requirements

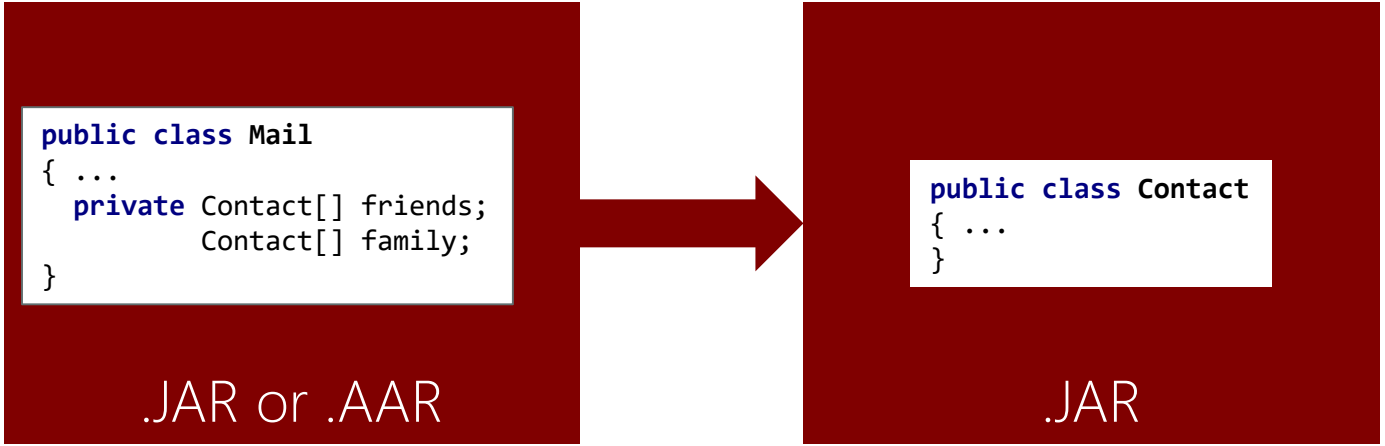


When you create a Bindings Library for this .JAR/.AAR…

…you do not have to include this .JAR in your project in any way

Runtime requirements

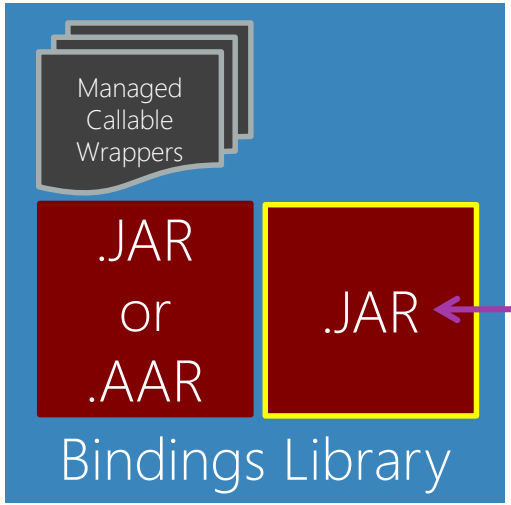
- A private-use reference .JARs must be available at runtime



This library needs to be compiled from Java bytecodes to Dalvik bytecodes and be available on the device at runtime

Reference .JAR embedding

- You can embed your private-use reference .JAR in your Bindings Library .DLL using the **`EmbeddedReferenceJar`** Build Action



`EmbeddedReferenceJar` will not generate bindings but will include it in the .DLL

Flash Quiz

Xamarin University

Flash Quiz

① You are **required** to include private-use reference .JARs in your Bindings Library project

a) True

b) False

Flash Quiz

① You are **required** include private-use reference .JARs in your Bindings Library project

a) True

b) False

Flash Quiz

② What does the `EmbeddedReferenceJar` Build Action do?

a) Embeds the .JAR and generates bindings

b) Embeds the .JAR but does not generate bindings

c) Does not embed the .JAR but does generate bindings

d) Does not embed the .JAR and does not generate bindings

Flash Quiz

② What does the `EmbeddedReferenceJar` Build Action do?

a) Embeds the .JAR and generates bindings

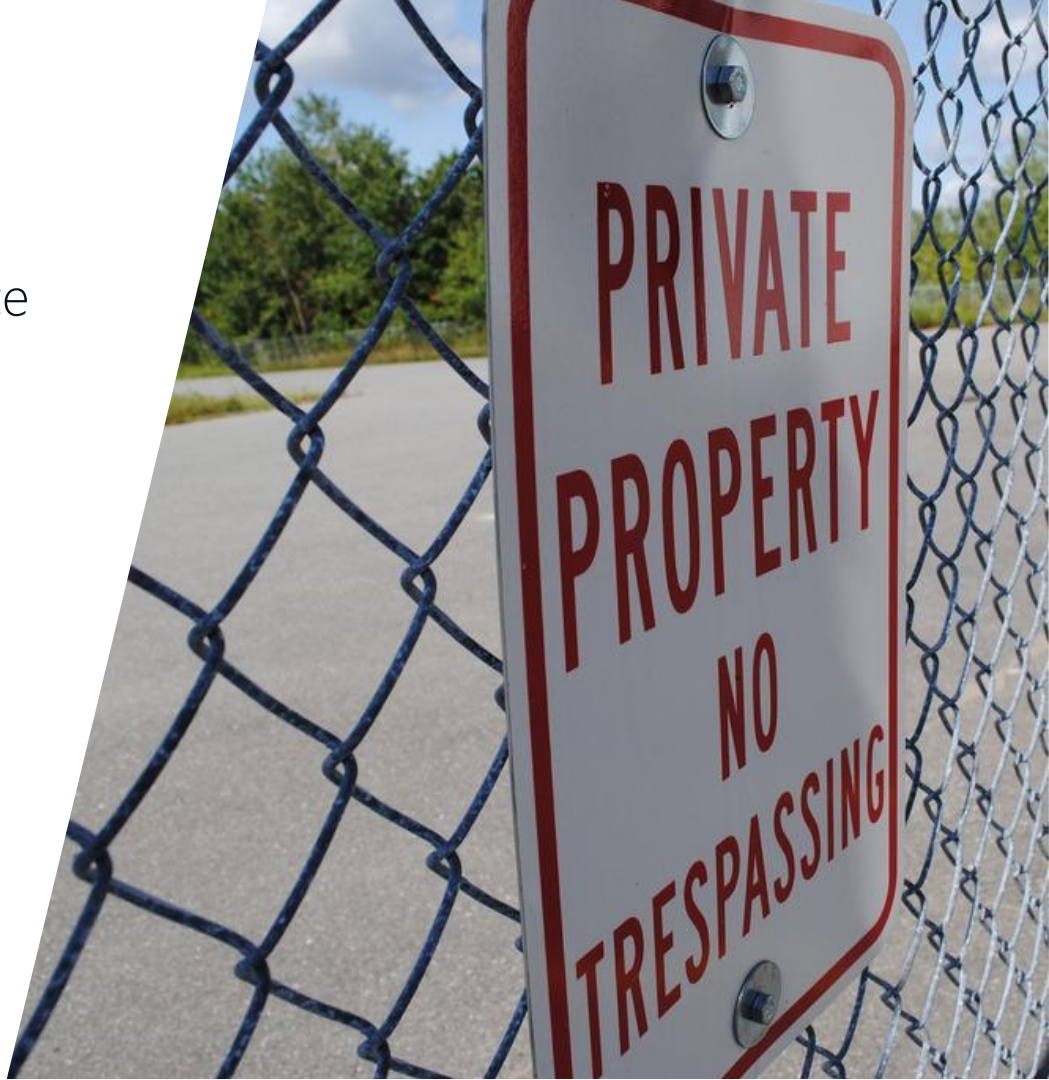
b) Embeds the .JAR but does not generate bindings

c) Does not embed the .JAR but does generate bindings

d) Does not embed the .JAR and does not generate bindings

Summary

1. Include a private-use reference .JAR in your Bindings Library



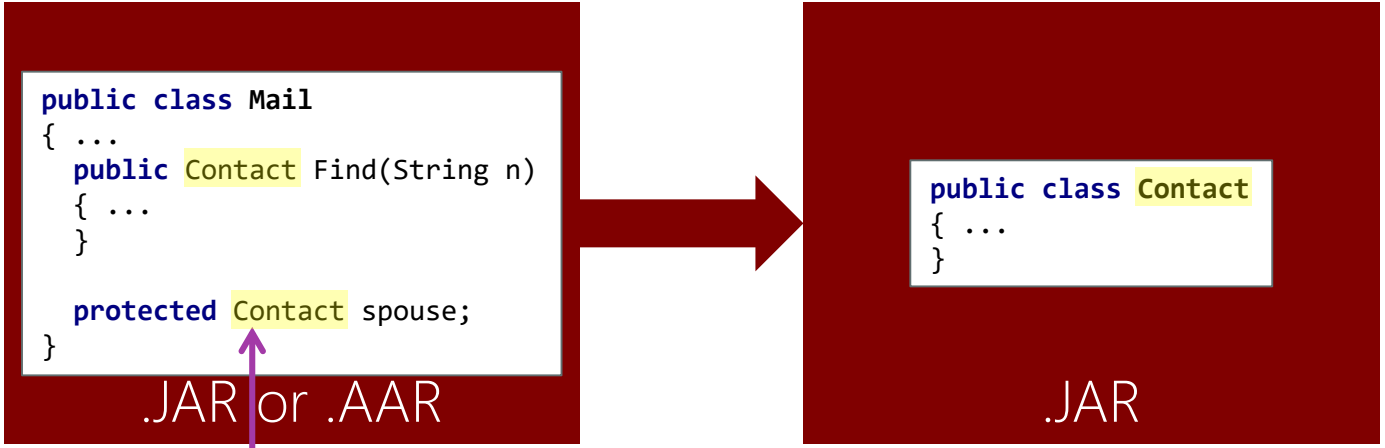
Handle a public-use reference .JAR

Tasks

1. Make a public-use reference .JAR available at compile time

What is a public-use reference .JAR?

- A reference .JAR is *public-use* if it exposes the **types from the reference .JAR** in public or protected members



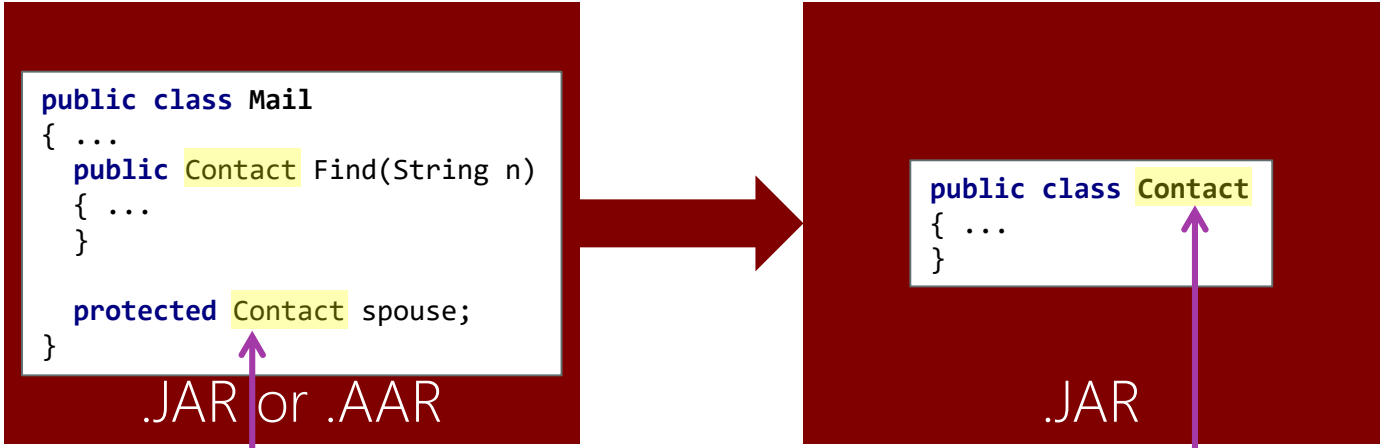
You return the type from a public method, have a protected field, etc.

This reference .JAR is public-use

Note: the term *public-use* is used in this course, but is not an official phrase

Bindings required

- You need Managed Callable Wrappers for the types in a public-use reference .JAR

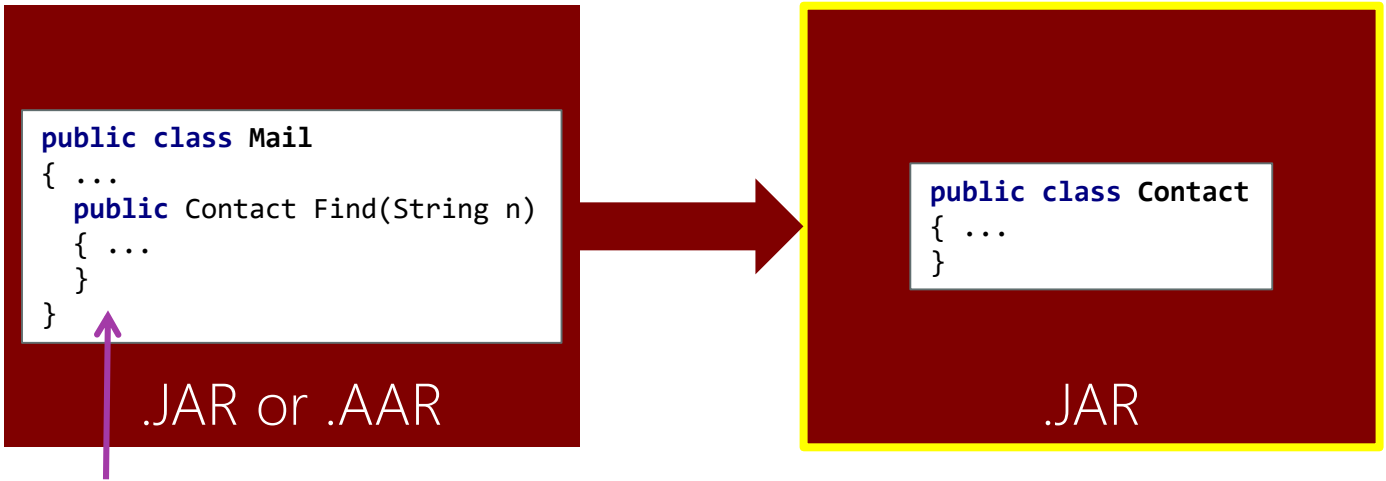


The **Contact** class is exposed to the client…

…so bindings are needed for **Contact**

Compile-time requirements

- **Reference .JARs** must be available at compile time, this is needed because the types are loaded into a JVM during binding generation



To load the **`Mail`** class, the JVM needs the **`Contact`** class to be available

Compile-time options [overview]

- There are three options for compile-time availability, you decide based on where/when you want to generate the bindings for the reference .JAR

Re-use: You already have a Bindings Library for the reference .JAR

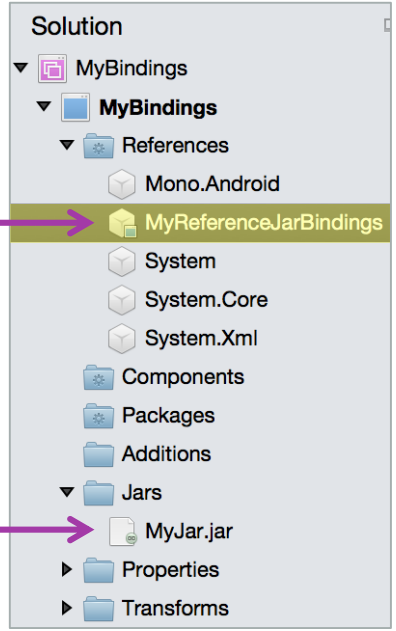
Create: You will make a Bindings Library for the reference .JAR but have not done so

Merge: You want to create the bindings for both libraries in this Bindings Library

Compile-time options [Re-use]

- An assembly reference for the .DLL containing the bindings for the reference .JAR satisfies the compile-time requirements, the Xamarin tooling reads the .JAR inside the .DLL

.DLL containing bindings for the reference .JAR, it has the reference .JAR inside



.JAR you are currently binding, it exposes types from the reference .JAR

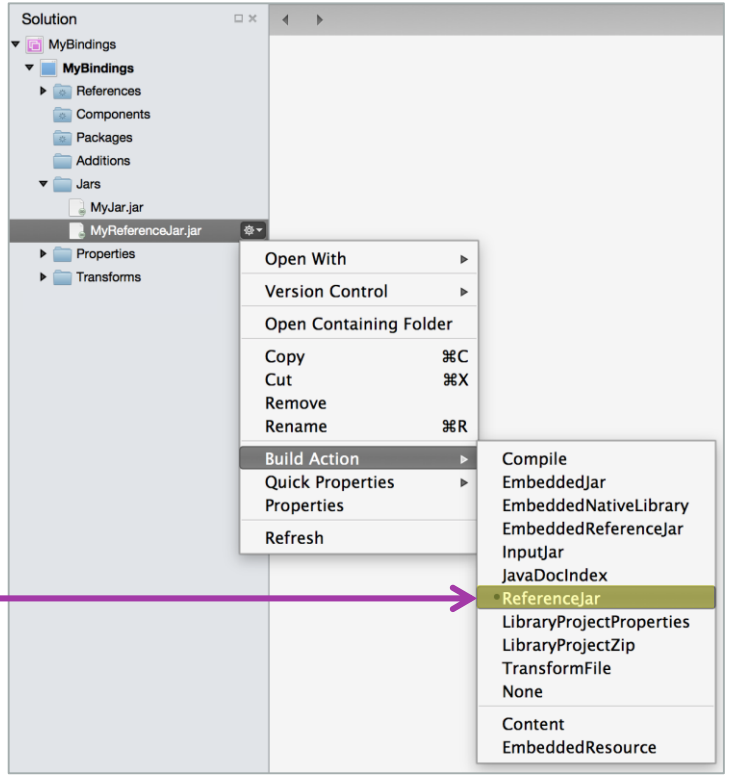
Re-use: You already have a Bindings Library for the reference .JAR

Compile-time options [Create]

- Including the reference .JAR in your Bindings Library project with the **`ReferenceJar`** Build Action satisfies the compile-time requirements

Create: You will make a Bindings Library for the reference .JAR but have not done so

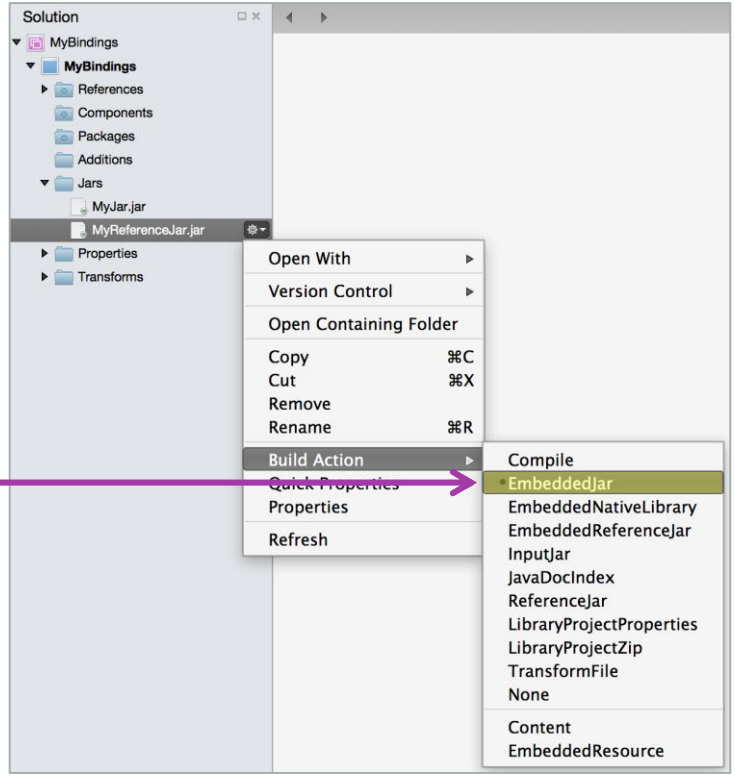
Do not create bindings, do not embed the reference .JAR in this .DLL



Compile-time options [Merge]

- Including the reference .JAR in your Bindings Library project with the **`EmbeddedJar`** Build Action satisfies the compile-time requirements

Create bindings and embed the reference .JAR in this .DLL



Merge: You want to create the bindings for both libraries in this Bindings Library

Flash Quiz

Xamarin University

Flash Quiz

① You must include public-use reference .JARs in your Bindings Library project

a) True

b) False

Flash Quiz

① You must include public-use reference .JARs in your Bindings Library project

a) True

b) False

Flash Quiz

② What does the `ReferenceJar` Build Action do?

a) Embeds the .JAR and generates bindings

b) Embeds the .JAR but does not generate bindings

c) Does not embed the .JAR but does generate bindings

d) Does not embed the .JAR and does not generate bindings

Flash Quiz

② What does the `ReferenceJar` Build Action do?

a) Embeds the .JAR and generates bindings

b) Embeds the .JAR but does not generate bindings

c) Does not embed the .JAR but does generate bindings

d) Does not embed the .JAR and does not generate bindings

Summary

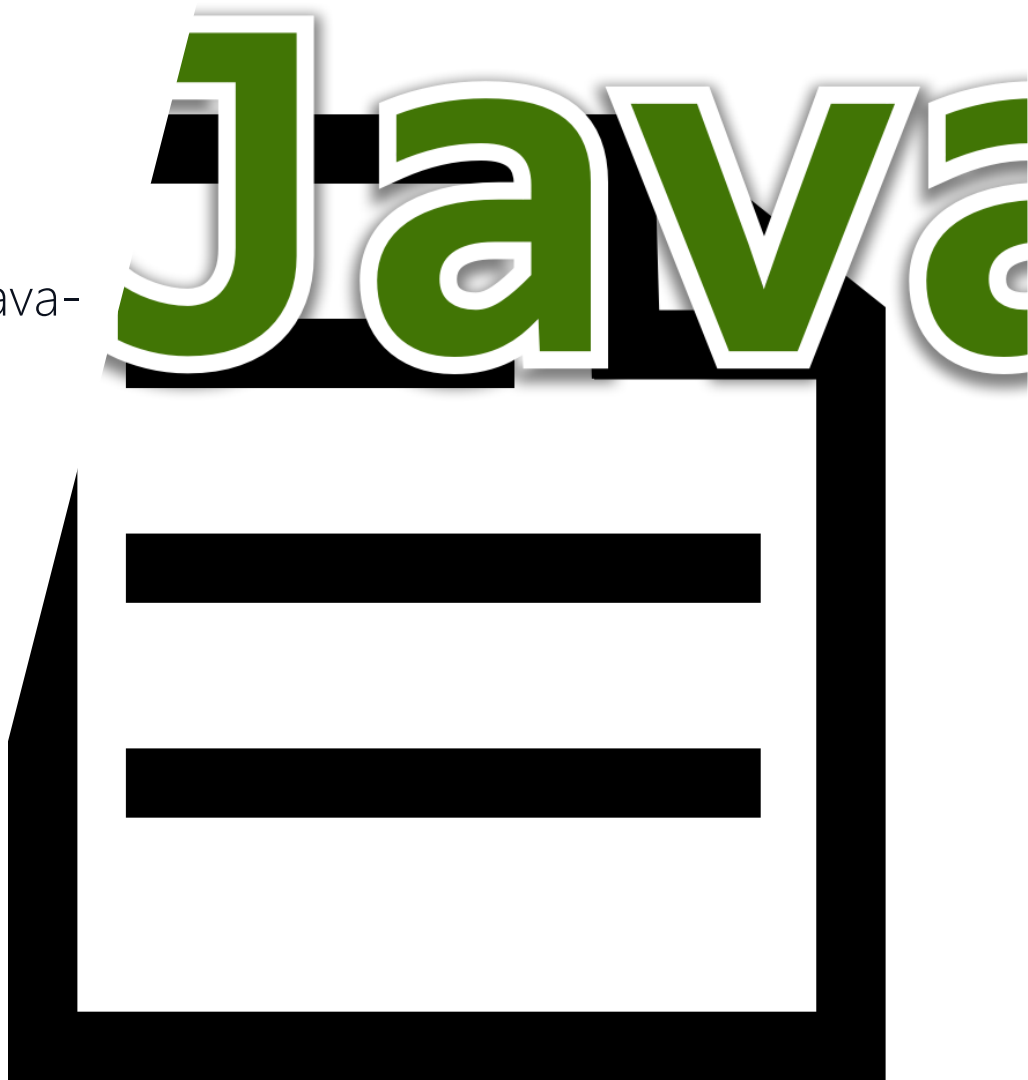
1. Make a public-use reference .JAR available at compile time

Examine how Java-language constructs are mapped to C#

Xamarin University

Tasks

1. Examine the C# Bindings for Java-language constructs



Motivation [language features]

- Some Java language constructs do not have a direct C# mapping

```
public interface Exportable
{
  void export(int format);

  public static final int PDF = 1;
  public static final int XPS = 2;
}
```

E.g. Java interfaces can contain constants which is not allowed in C#

Motivation [patterns]

- Some Java patterns are mapped to C# language features

```
public class Contact
{ ...
  public String getName()          { ... }
  public void   setName(String n) { ... }
}
```

E.g. C# would use properties, not get/set methods

Motivation [conventions]

- Java and C# use different conventions for capitalization and naming

```
public class Contact
{ ...
  public String send(String message)
  {
    ...
  }
}
```

E.g. C# would use an uppercase first letter

Individual Exercise

Examine mappings for various Java constructs

Xamarin University

Binding namespaces

- Java packages map to C# namespaces with uppercase first letters

```
package com.mycompany.mypackage;
```

Package

```
namespace Com.Mycompany.Mypackage
{
   ...
}
```

Namespace

Binding classes

- Java classes map to C# classes

```
public class Contact
{
   ...
}
```

Java Class

```
public class Contact : Java.Lang.Object
{
   ...
}
```

C# Base type is Java's **Object** class

Binding interfaces

- Java interfaces map to C# interfaces with an uppercase '**I**' added to the name and a class for constants (if needed)

```
public interface Exportable
{
  void export(int format);

  public static final int PDF = 1;
  public static final int XPS = 2;
}
```

Java Interface with methods and constants

```
public interface IExportable
{
  void Export(int p0);
}
```

```
public abstract class Exportable
{
  public const int Pdf = 1;
  public const int Xps = 2;
}
```

C# Interface for methods, class for constants

Binding instance methods

- Java instance methods map to C# **`virtual`** methods since Java methods use virtual dispatch by default

```
public class Contact
{ ...
  public String send(String message)
  {
    ...
  }
}
```

Java Instance method

```
public class Contact : Java.Lang.Object
{ ...
  public virtual string Send(string p0)
  {
    ...
  }
}
```

C# Virtual instance method, parameter names are not preserved

Binding static methods

- Java **static** methods map to C# **static** methods

```
public class Contact
{ ...
  public static String format(String value)
  {
    ...
  }
}
```

Static method

```
public class Contact : Java.Lang.Object
{ ...
  public static string Format(string p0);
  {
    ...
  }
}
```

Static method

Binding get/set methods

- Java get/set methods map to C# properties

```
public class Contact
{ ...
  public String getName()          { ... }
  public void   setName(String n) { ... }

  public int getAge() { ... }

  public void setRate(double r) { ... }
}
```

```
public class Contact : Java.Lang.Object
{ ...
  public virtual string Name { get; set; }


  public virtual int Age { get; }

  public virtual void SetRate(double p0);
}
```

Java Properties created for get/set and get-only, but not set-only

C# Properties for **Name** and **Age**, method for **Rate**

Binding fields

- Java fields map to C# properties

```
public class Contact
{ ...
  public int address;
}
```

Java

Public field

```
public class Contact : Java.Lang.Object
{ ...
  public int Address { get; set; }
}
```

C#

Read/write property

Binding listeners

- Java set/add methods with a specific signature map to C# events

```
public class Employee
{ ...
  public void addSalaryListener(SalaryListener subscriber) { ... }
  public void setSalaryListener(SalaryListener subscriber) { ... }
}
```

Java

Must have **void** return

Must have **add** or **set** prefix and **Listener** suffix

Must have a single parameter of an interface type

```
public event EventHandler<SalaryEventArgs> SalaryEvent;
```

C#

Binding generates a delegate, an event-args class, and an event

Binding static nested classes

- Java static nested classes map to C# nested classes

```
public class List
{ ...
  public static class Node
  {
  }
}
```

Java Static nested class

```
public class List : Java.Lang.Object
{ ...
  public class Node : Java.Lang.Object
  {
  }
}
```

C# Nested class

Binding inner classes

- Java inner classes map to C# nested classes with a constructor that requires an instance of the outer class

```
public class List
{ ...
  public class Iterator
  {
  }
}
```

Java

Inner class

```
public class List : Java.Lang.Object
{ ...
  public class Iterator : Java.Lang.Object
  {
    public Iterator(List __self) { ... }
  }
}
```

C#

Must pass instance of outer class

Bound access levels

- Java **public** and **protected** members get C# bindings by default

```
public class Contact
{ ...
  public    int myFieldPublic;
  protected int myFieldProtected;

  private   int myFieldPrivate;
            int myFieldPackage;
}
```

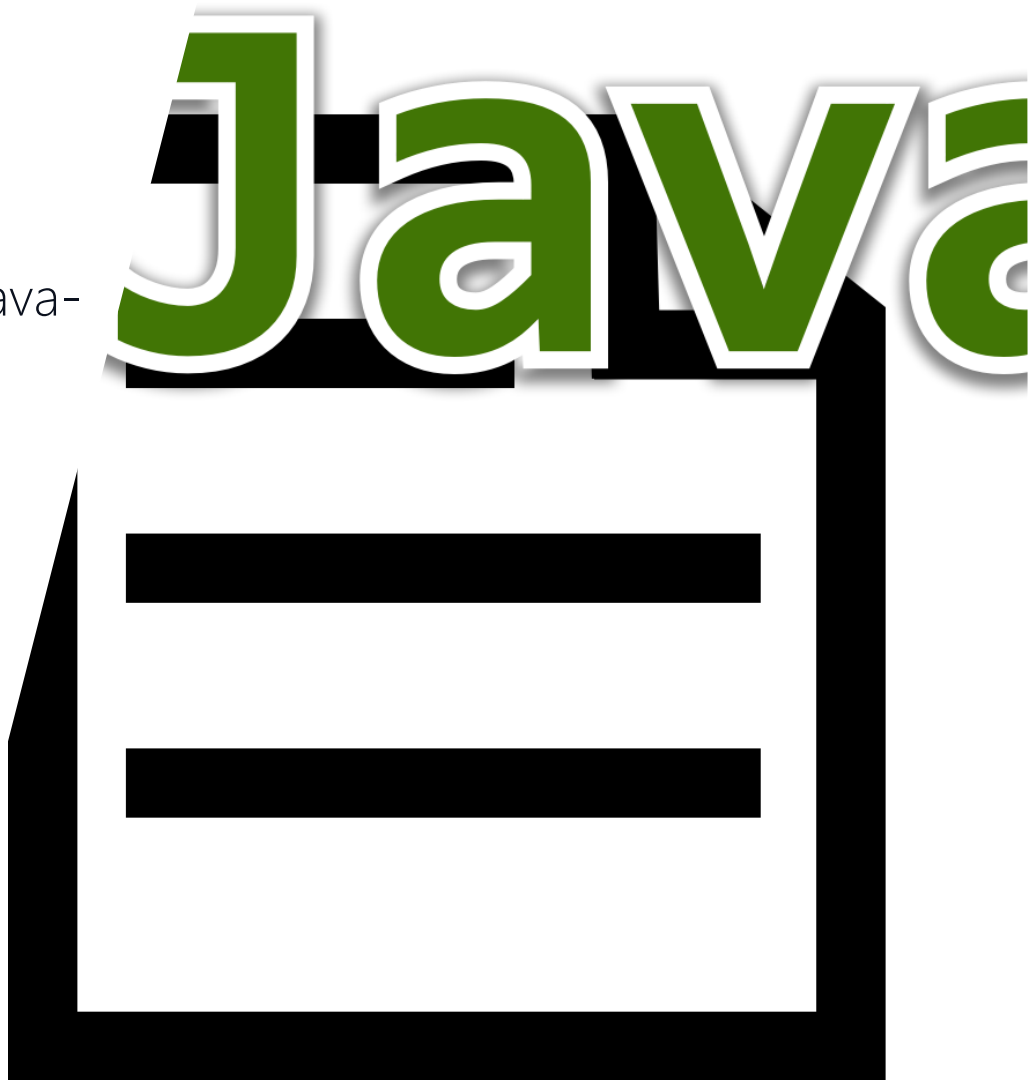
```
public class Contact : Java.Lang.Object
{ ...
  public    int MyFieldPublic    { get; set; }
  protected int MyFieldProtected { get; set; }
}
```

Java **private** and **package** members not bound

C# **public** and **protected** member are bound, access level is preserved

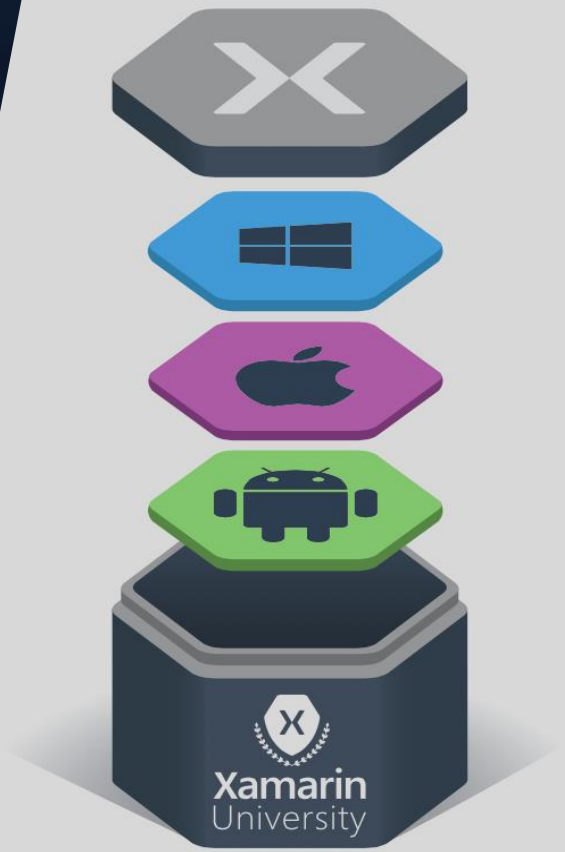
Summary

1. Examine the C# Bindings for Java-language constructs



Thank You!

Please complete the class survey in your profile:
university.xamarin.com/profile



Microsoft