Lecture 20: Inheritance and Polymorphism

08/10/22

Upcoming

- A5 Resubmission due Wedneday 8/10 @ 11:59pm
- A7 due Thursday 8/11 @ 11:59pm
- *A8* due **Tuesday** 8/16 @ 11:59pm
  - Cannot be resubmitted!
  - Late days allowed, but the last day of IPL is Wednesday, 8/17

Final Exam

Resources posted:
https://courses.cs.washington.edu/courses/cse143/22su/exams.shtml

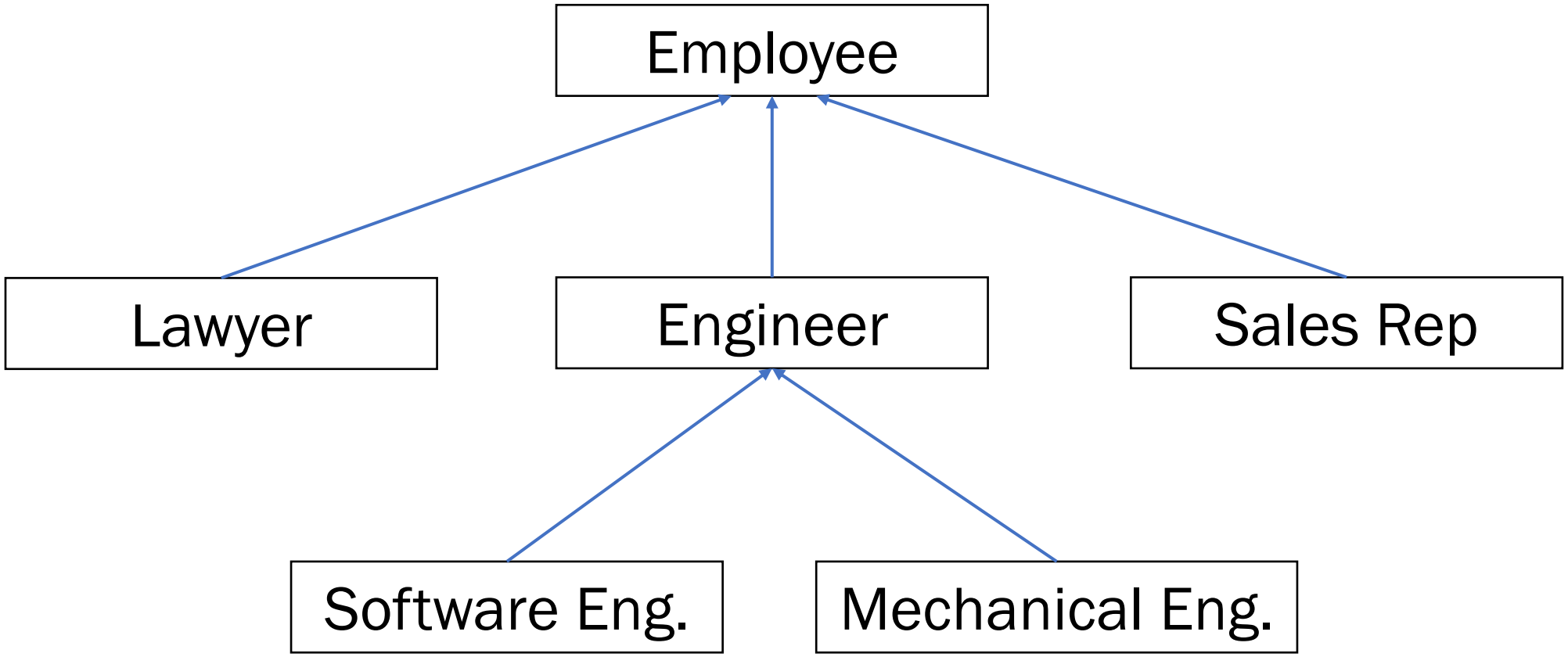
Split into two parts:

- Thursday 8/18: Final Exam part 1 in your section
- Friday 8/19: Final Exam part 2, 10:50 - 11:50am

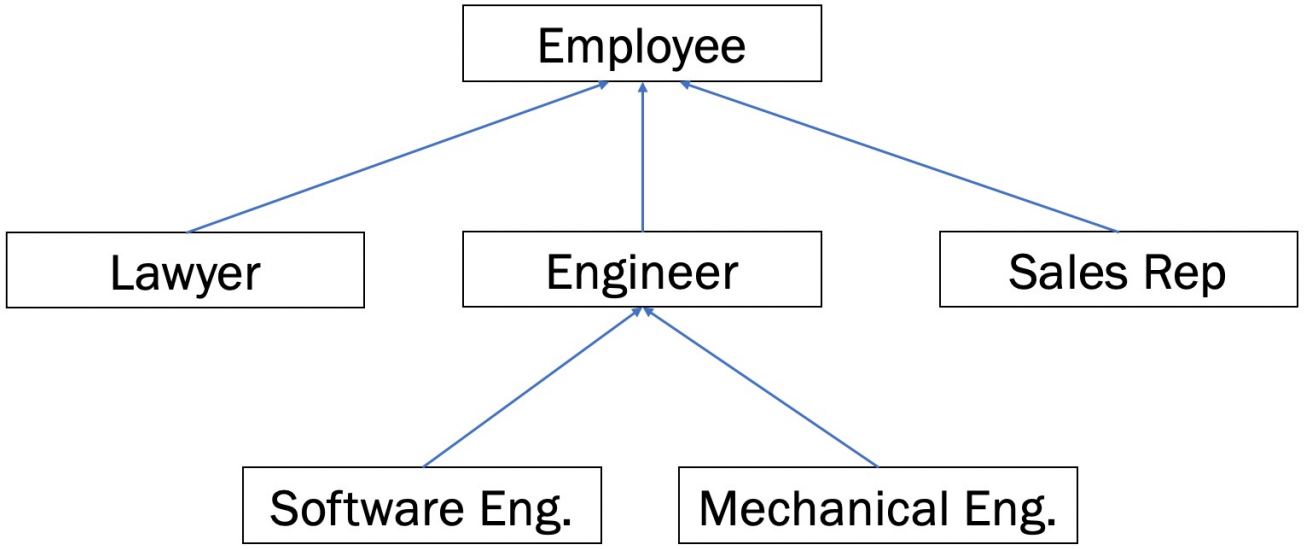
Optional review session:

- Friday 8/12 from 2:20 - 3:20pm in GUG 220

Inheritance

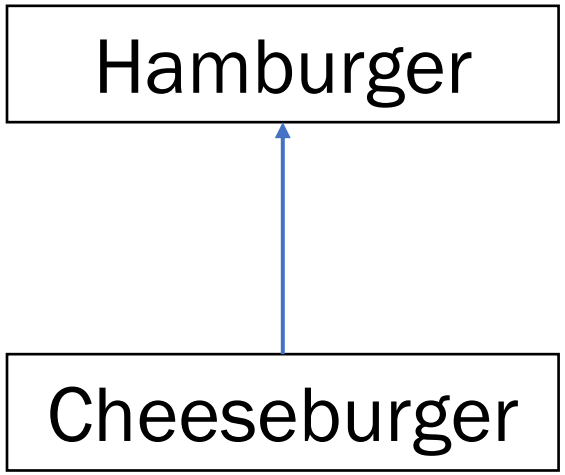


Inheritance in Java




```
public class Employee {
    ...
}

public class Engineer extends Employee {
    ...
}
```

Superclass vs. Subclass



Polymorphism

```
public class Employee {
    public void completeTask() {
        System.out.println("starting task:");
        doWork();
        System.out.pritnln("done!");
    }

    public void doWork() {
        System.out.println("shredding paper...");
    }
}

public class Engineer extends Employee {
    public void doWork() {
        System.out.println("beep boop...");
    }
}
```

```
Engineer e = new Engineer();
e.completeTask();
```

What is the output of the above code?

Inheritance

- **inheritance**: Forming new classes based on existing ones.
  - a way to share/**reuse code** between two or more classes
  - **superclass**: Parent class being extended.
  - **subclass**: Child class that inherits behavior from superclass.
  - **is-a relationship**: Each object of the subclass also "is a(n)" object of the superclass and can be treated as one.

Poll Everywhere

pollev.com/cse143

```
public class A {
  public void m1() {
    m2();
    System.out.println("A1");
  }

  public void m2() {
    System.out.println("A2");
  }
}

public class B extends A {
  public void m2() {
    System.out.println("B2");
  }
}
```

```
B b = new B();
b.m1();
```

What is the output of the above code?

- `A2 / A1`
- `B2 / A1`
- Some kind of error

```
public class Employee {
    public void doWork() {
        System.out.println("shredding paper...");
    }
}

public class Engineer extends Employee {
    public void mystery() {
        doWork();
        super.doWork();
    }

    public void doWork() {
        System.out.println("beep boop...");
    }
}


public class SoftwareEngineer extends Engineer {
    public void doWork() {
        System.out.println("Hello World!");
    }
}
```

```
SoftwareEngineer e2 = new
        SoftwareEngineer();
e2.mystery();
```

What is the output of the above code?

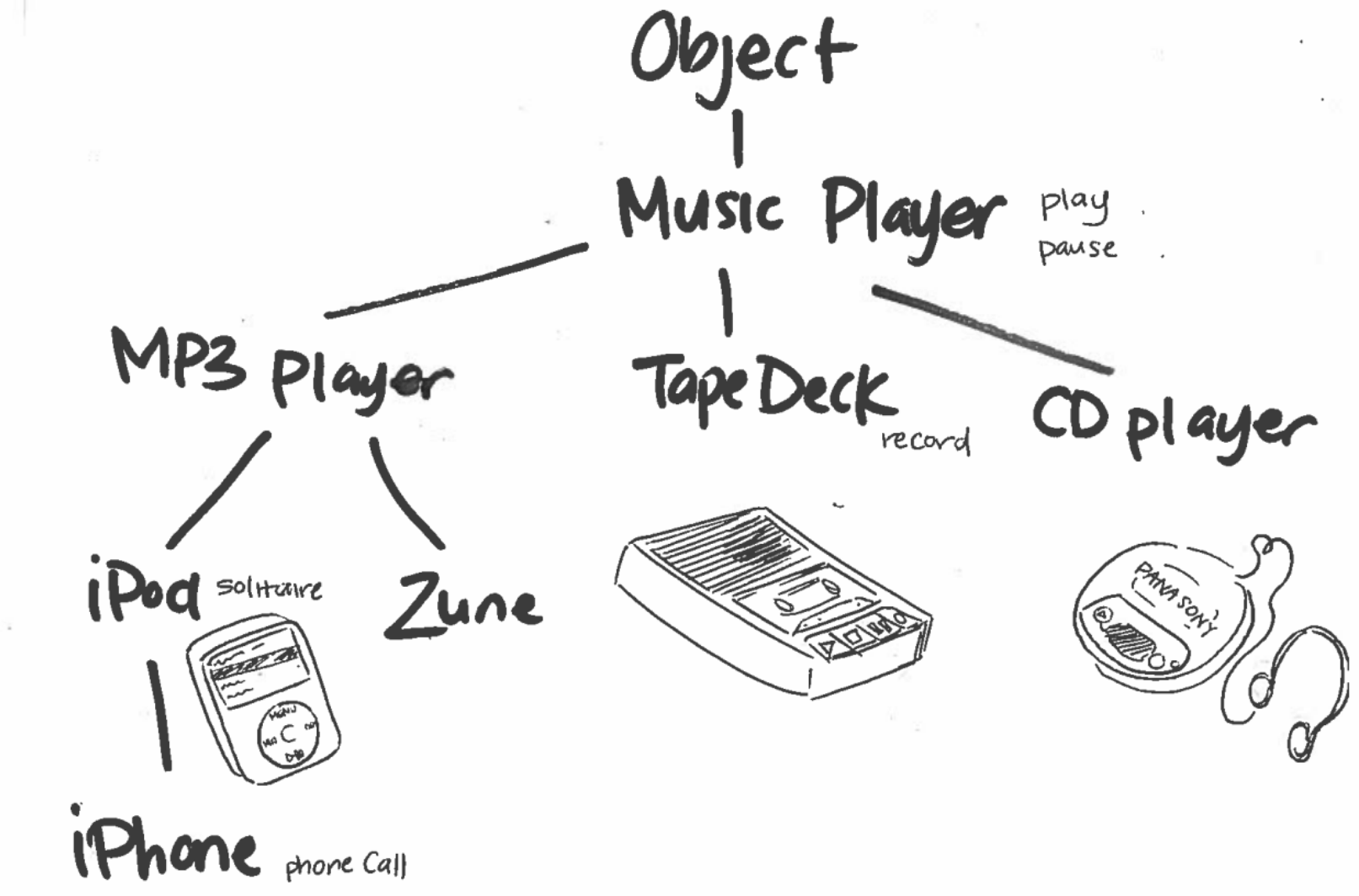
Where we're going

- Practice with inheritance and polymorphism
- Understanding Java's type system
  - What happens when you using casting with objects?
  - What is and isn't possible for the compiler to check?

- Motivation: what does this line of code mean?

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

Example: Music Players



Java Type System

Which of these lines of code work?

```
MP3Player a1 = new MP3Player();
IPhone a2 = new IPhone();

MP3Player a3 = new IPhone();
IPhone 34 = new MP3Player();
```

Java Type System

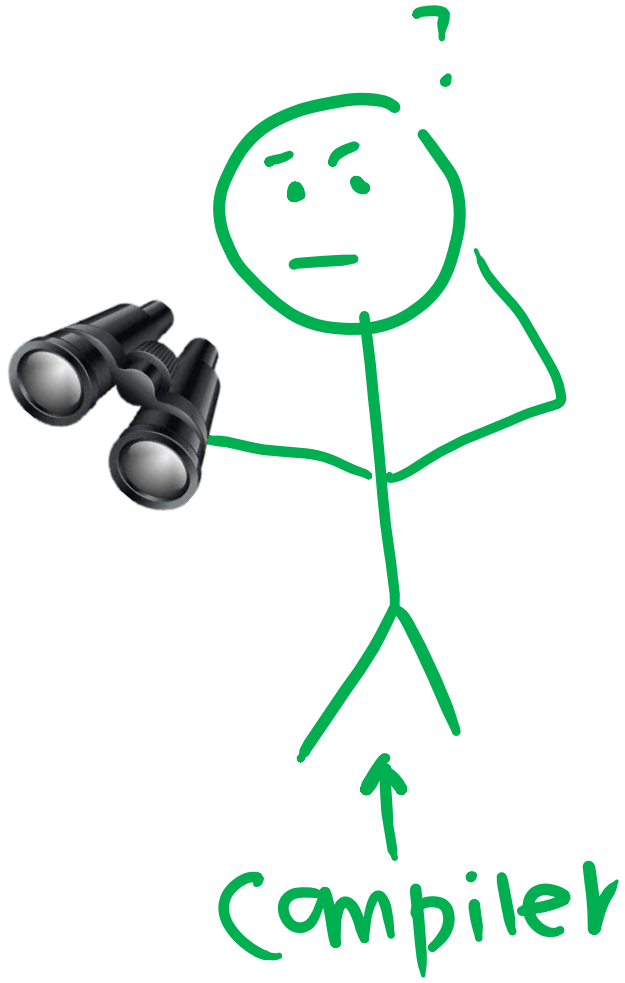
```
DeclaredType x = new ActualType();
```

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

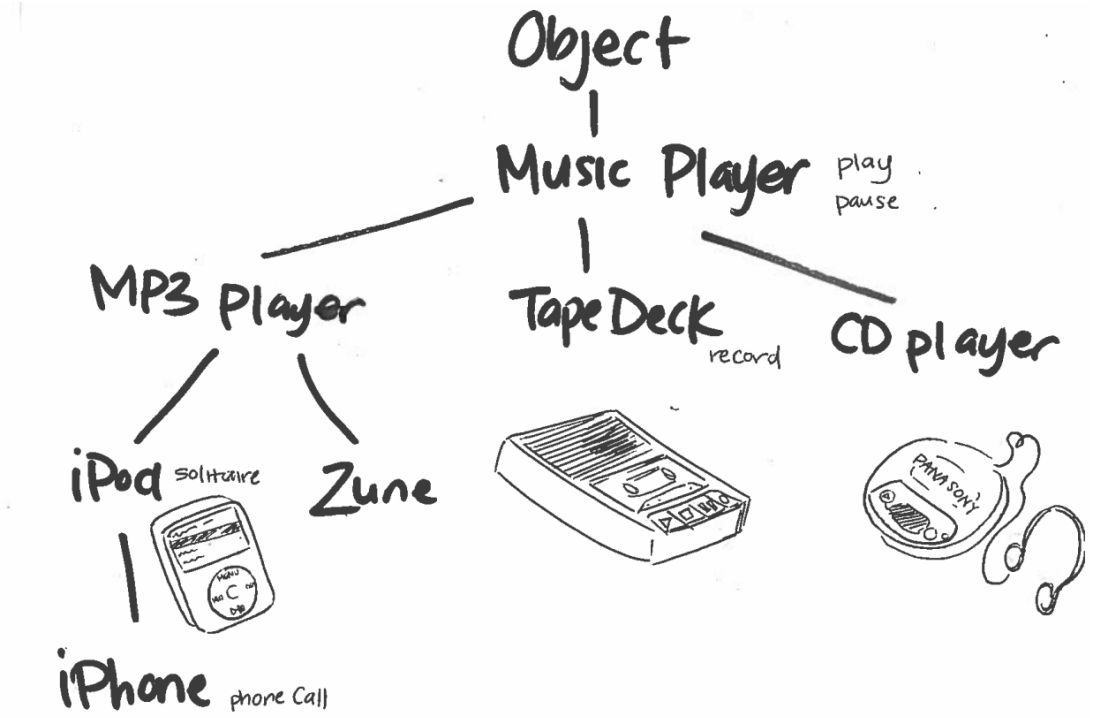
```
Scanner input = new Scanner();
```

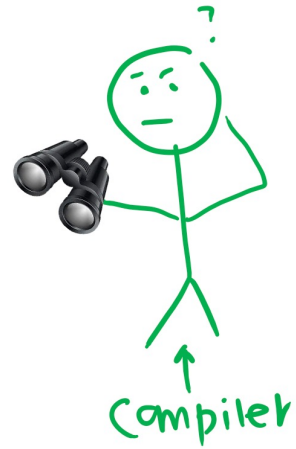
Compile Time vs. Runtime

Compile Time vs. Runtime



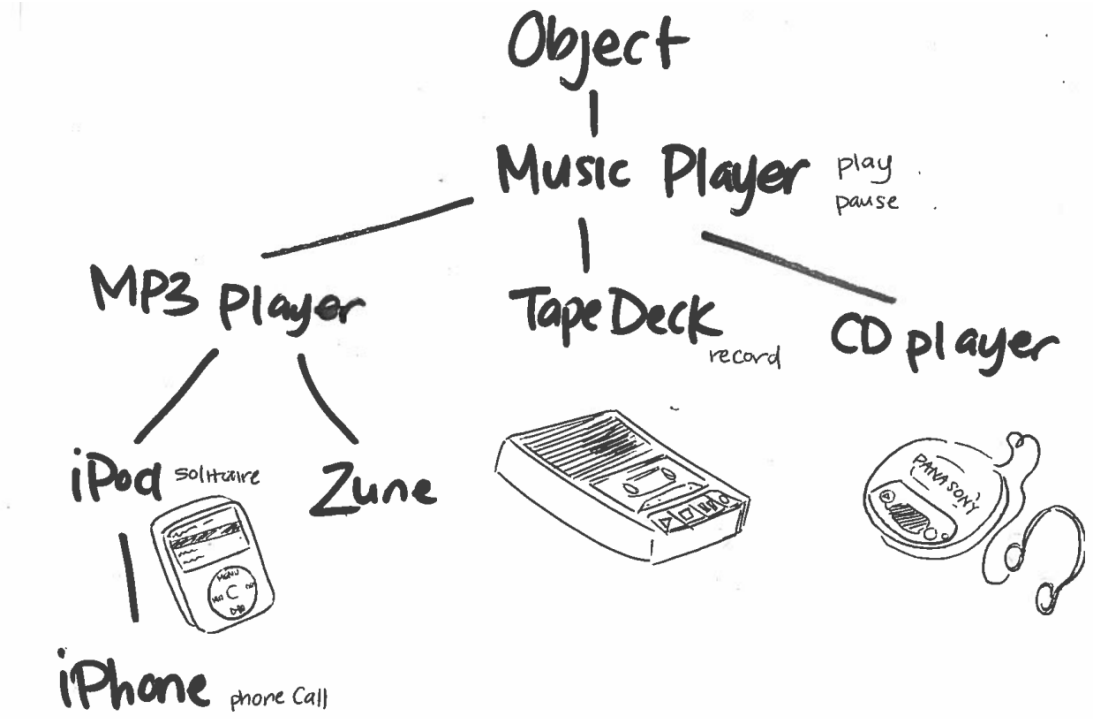
```
MusicPlayer p = new iPhone();
p.play();


p.phoneCall();


((IPhone)p).phoneCall();
```

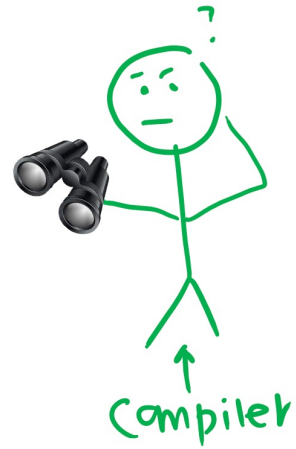




```
MP3Player p3 = new Zune();
((IPhone)p3).phoneCall();
```

```
((IPhone)p3).play();
```

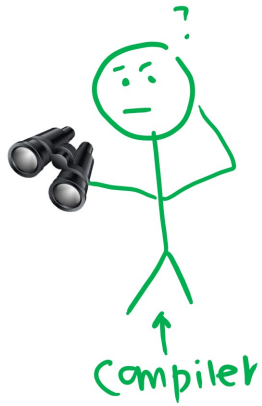


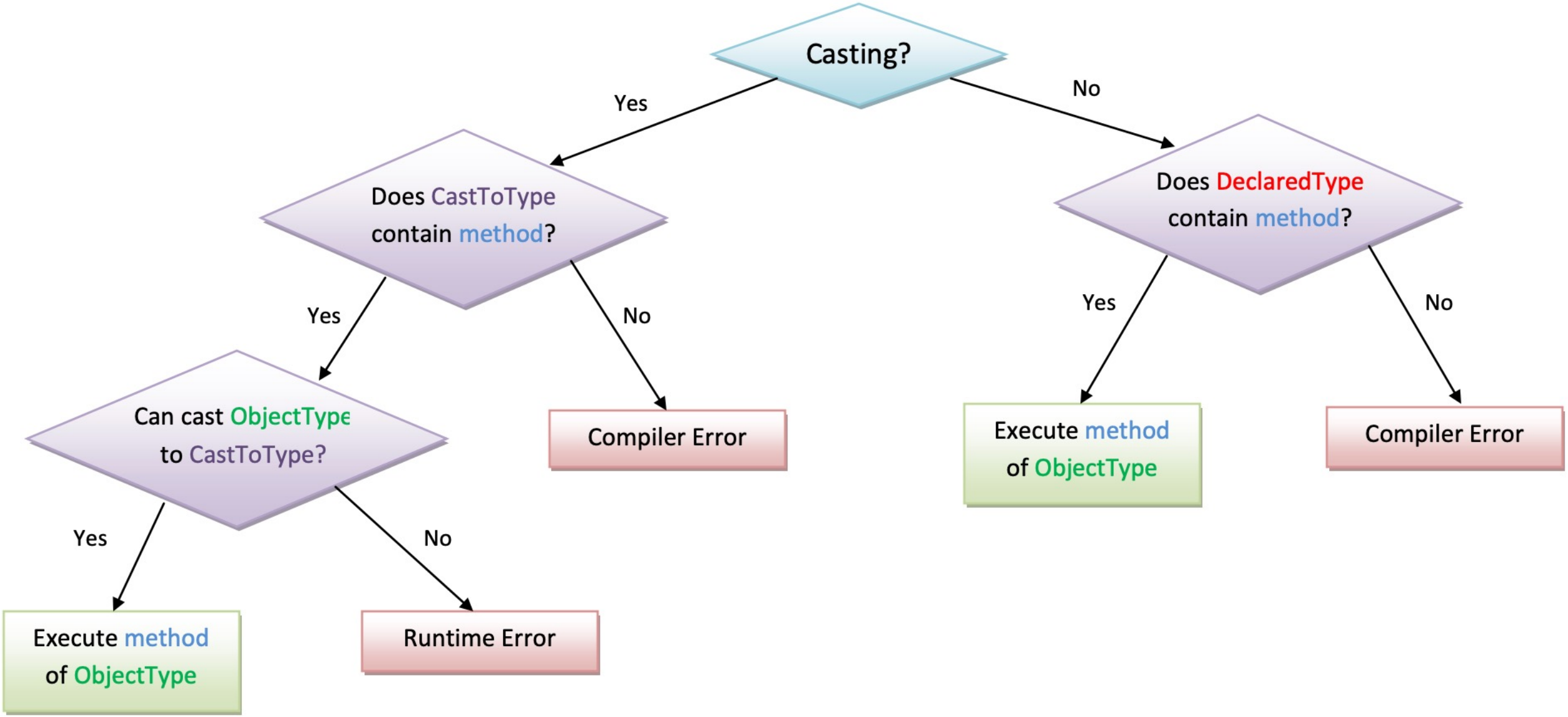


Compiler error vs. Runtime error summary

Steps:

1. Compiler – looks at the DeclaredType / cast type. Are we guaranteed that this type has the right method?
2. Runtime – did we meet our promise? Is the ActualType a subtype of the DeclaredType?
3. Run the method on the ActualType – the declared / cast type doesn't matter




Casting?
Yes
No
Does CastToType contain method?
Does DeclaredType contain method?
Yes
No
Yes
No
Can cast ObjectType to CastToType?
Compiler Error
Execute method of ObjectType
Compiler Error
Yes
No
Execute method of ObjectType
Runtime Error

```
public class MusicPlayer {
  public void m1() {
    System.out.println("MusicPlayer1");
  }
}

public class TapeDeck extends MusicPlayer {
  public void m3() {
    System.out.println("TapeDeck3");
  }
}

public class IPod extends MusicPlayer {
  public void m2() {
    System.out.println("IPod2");
    m1();
  }
}
```

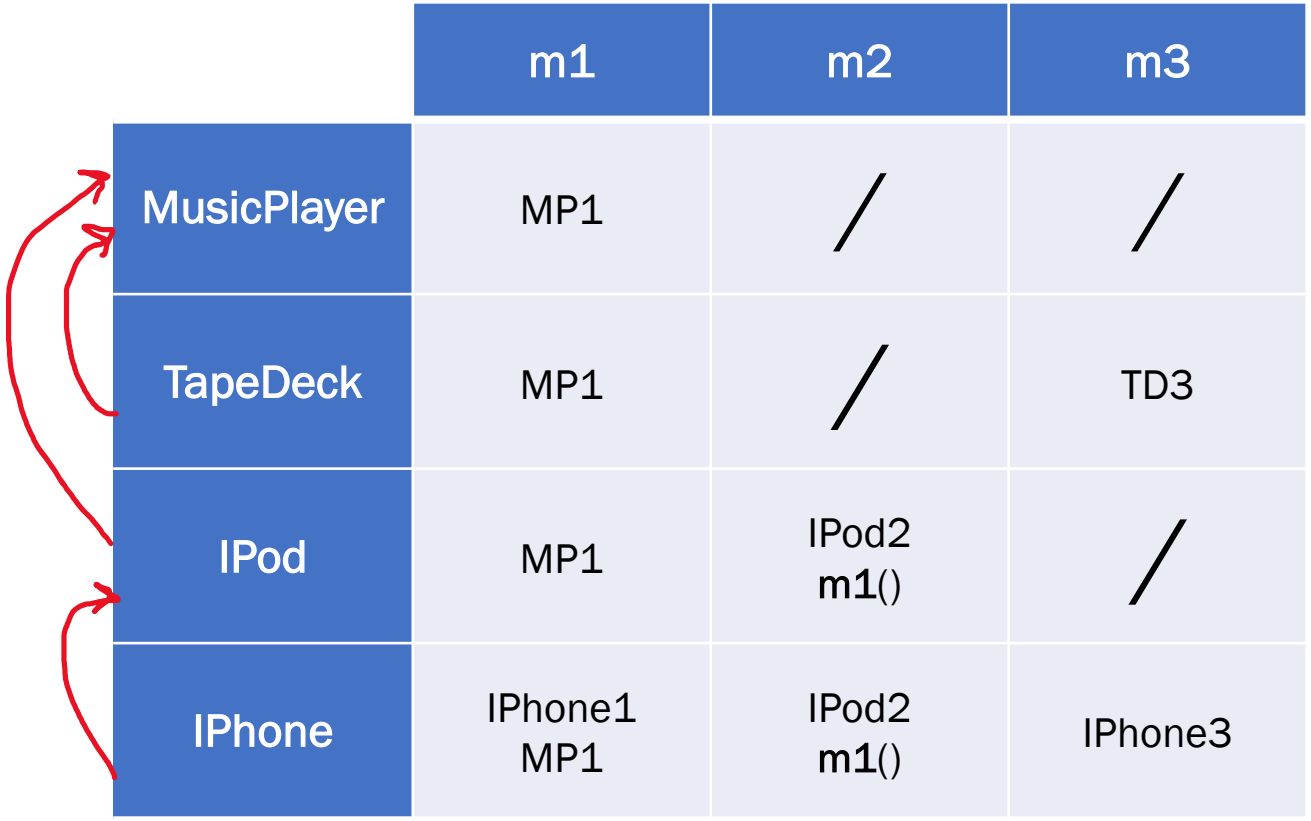
```
public class IPhone extends IPod {
  public void m1() {
    System.out.println("IPhone1");
    super.m1();
  }
  public void m3() {
    System.out.println("IPhone3");
  }
}
```

```
MusicPlayer var1 = new TapeDeck();
MusicPlayer var2 = new IPod();
MusicPlayer var3 = new IPhone();


var1.m1();

var3.m1();

var3.m2();
```

```
public class MusicPlayer {
  public void m1() {
    System.out.println("MusicPlayer1");
  }
}

public class TapeDeck extends MusicPlayer {
  public void m3() {
    System.out.println("TapeDeck3");
  }
}

public class IPod extends MusicPlayer {
  public void m2() {
    System.out.println("IPod2");
    m1();
  }
}
```

```
public class IPhone extends IPod {
  public void m1() {
    System.out.println("IPhone1");
    super.m1();
  }
  public void m3() {
    System.out.println("IPhone3");
  }
}
```

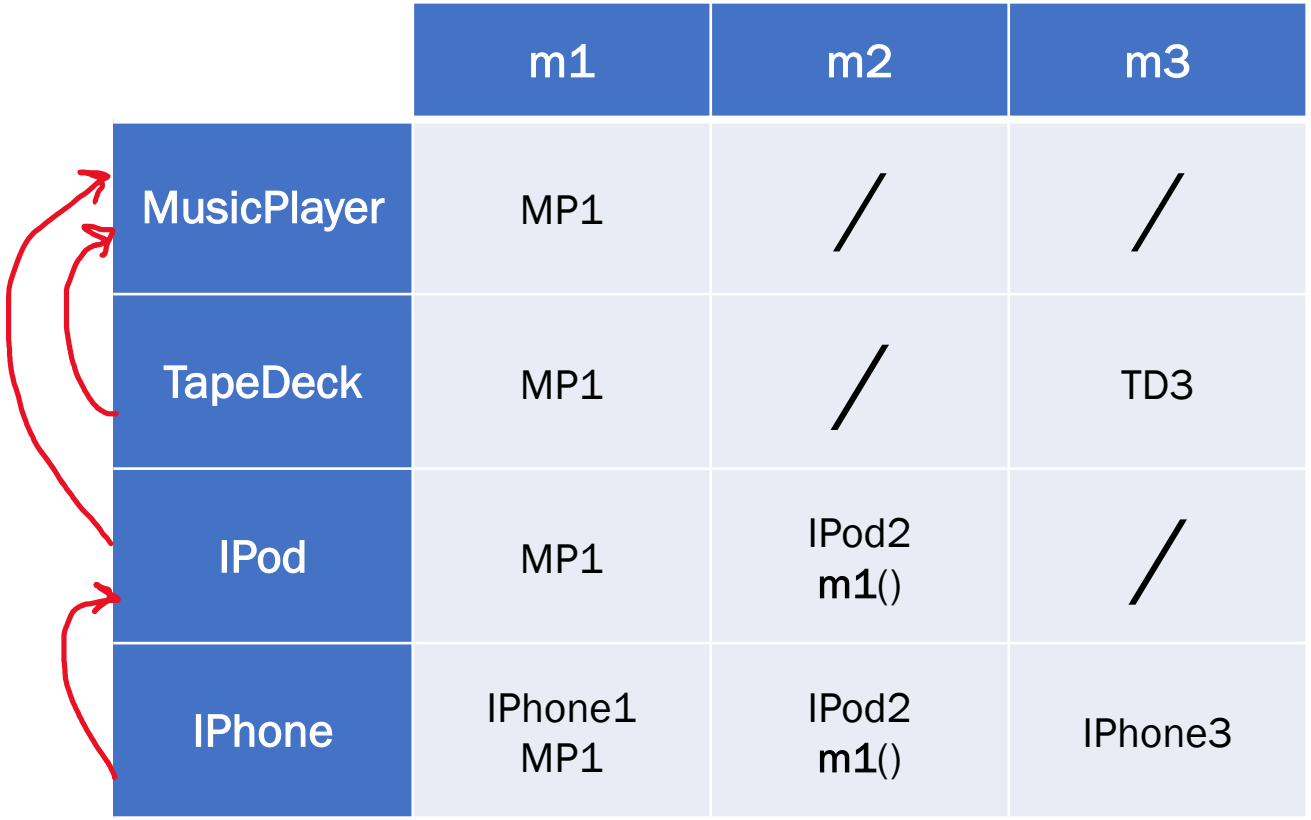
	m1	m2	m3
MusicPlayer			
TapeDeck			
IPod			
IPhone			



	m1	m2	m3
MusicPlayer	MP1	/	/
TapeDeck	MP1	/	TD3
IPod	MP1	IPod2 **m1**()	/
IPhone	IPhone1 MP1	IPod2 **m1**()	IPhone3

```
MusicPlayer var1 = new TapeDeck();
MusicPlayer var2 = new IPod();
MusicPlayer var3 = new IPhone();
IPod var4 = new IPhone();
Object var5 = new IPod();
Object var6 = new MusicPlayer();
```

```
var3.m1();
var4.m2();
var3.m2();
var5.m1();
((IPhone) var2).m1();
```



	m1	m2	m3
MusicPlayer	MP1	/	/
TapeDeck	MP1	/	TD3
IPod	MP1	IPod2 m1()	/
IPhone	IPhone1 MP1	IPod2 m1()	IPhone3

```
MusicPlayer var1 = new TapeDeck();
MusicPlayer var2 = new IPod();
MusicPlayer var3 = new IPhone();
IPod var4 = new IPhone();
Object var5 = new IPod();
Object var6 = new MusicPlayer();
```

```
var3.m1();
```

IPhone1 / MusicPlayer1

```
var4.m2();
```

IPod2 / IPhone1 / MusicPlayer1

```
var3.m2();
```

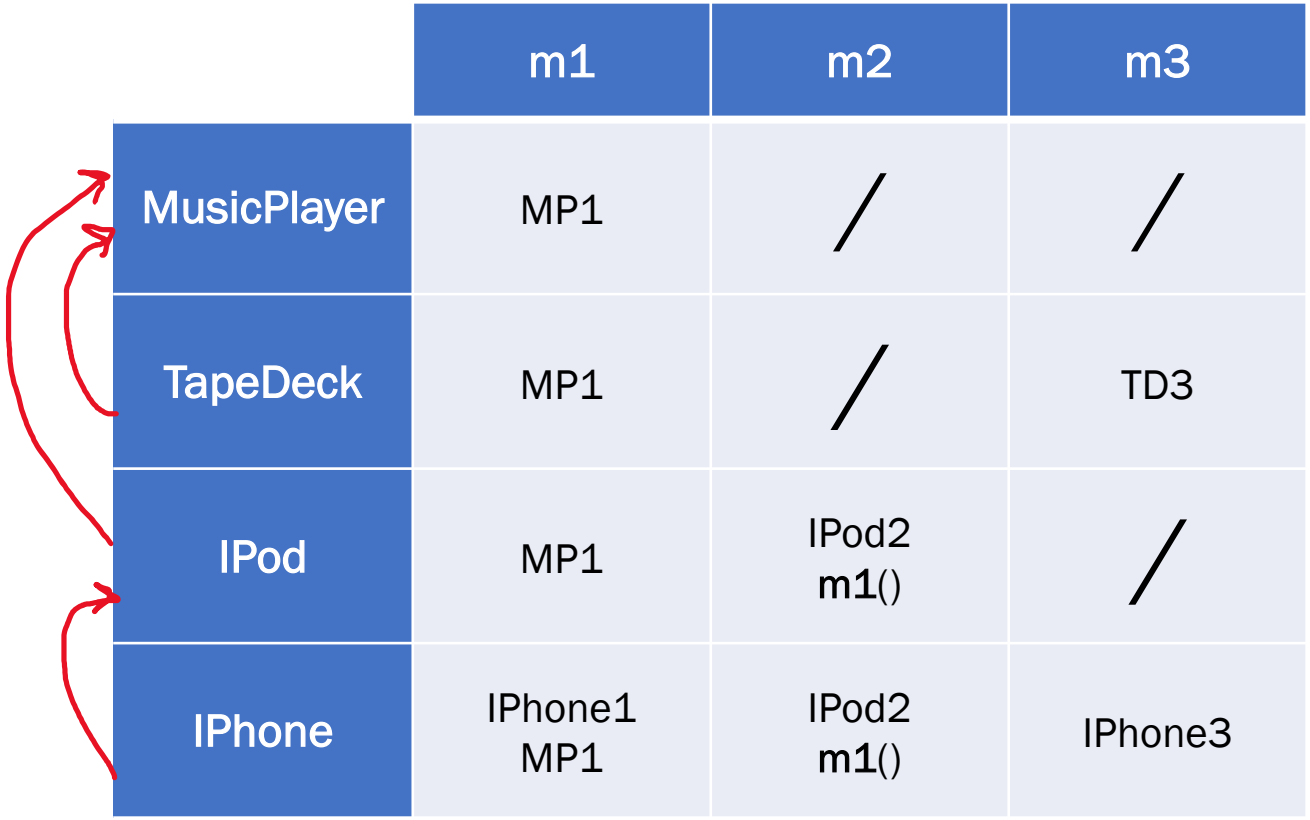
Compiler Error (CE)

```
var5.m1();
```

Compiler Error (CE)

```
((IPhone) var2).m1();
```

Runtime Error (RE)



	m1	m2	m3
MusicPlayer	MP1	/	/
TapeDeck	MP1	/	TD3
IPod	MP1	IPod2 **m1**()	/
IPhone	IPhone1 MP1	IPod2 **m1**()	IPhone3

```
MusicPlayer var1 = new TapeDeck();
MusicPlayer var2 = new IPod();
MusicPlayer var3 = new IPhone();
IPod var4 = new IPhone();
Object var5 = new IPod();
Object var6 = new MusicPlayer();
```

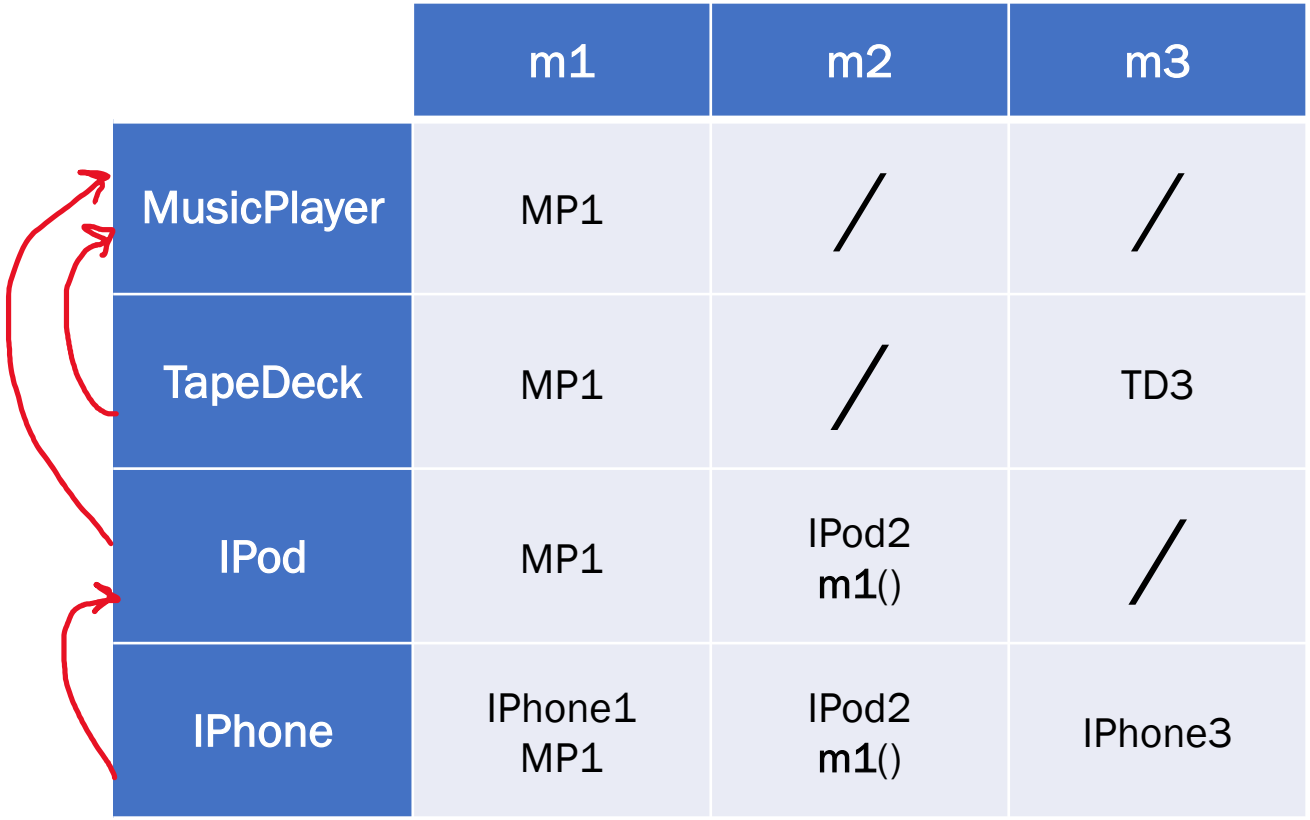
```
var1.m1();
```

```
((TapeDeck) var1).m2();
```

```
((IPod) var3).m2();
```

```
((TapeDeck) var3).m2();
```

```
((Iphone) var5).m2();
```



	m1	m2	m3
MusicPlayer	MP1	/	/
TapeDeck	MP1	/	TD3
IPod	MP1	IPod2 m1()	/
IPhone	IPhone1 MP1	IPod2 m1()	IPhone3

```
MusicPlayer var1 = new TapeDeck();
MusicPlayer var2 = new IPod();
MusicPlayer var3 = new IPhone();
IPod var4 = new IPhone();
Object var5 = new IPod();
Object var6 = new MusicPlayer();
```

```
var1.m1();
```

MusicPlayer1

```
((TapeDeck) var1).m2();
```

Compiler Error (CE)

```
((IPod) var3).m2();
```

IPod2 / IPhone1 / MusicPlayer1

```
((TapeDeck) var3).m2();
```

Compiler Error (CE)

```
((Iphone) var5).m2();
```

Runtime Error (RE)