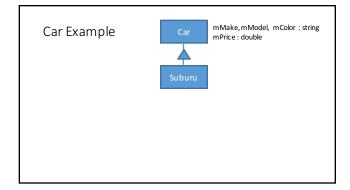
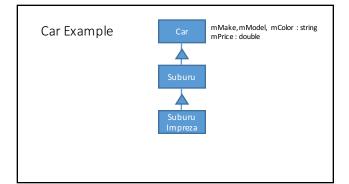
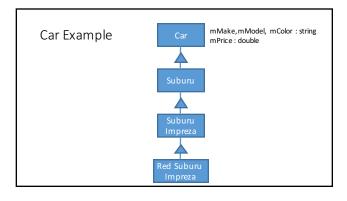
CS 31: Introduction To Computer Science I Howard A Stahl	
Car Example Car Example Car Example Make, mModel, mColor : string mPrice : double	
Car Example Car Example Car Example mMake,mModel, mColor : string mPrice : double	

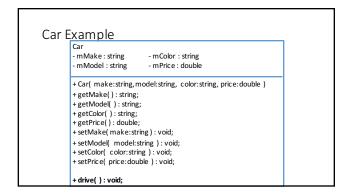


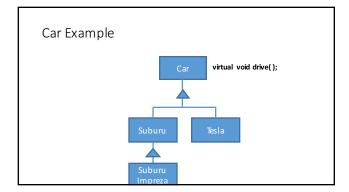


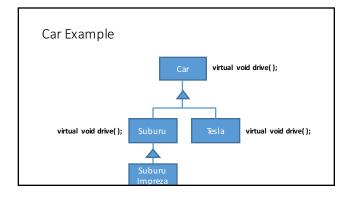


Inheritance A Structural Concept That Supports The "IS-A" Relationship An Additive Models Do not Repeat Visuals Leves Of An Oxion Parrot Class Sub Class Inherited Behavior Subclass Com "Covertife" The Behavior Of Their Parret Class By Providing A Different Implementation of A Certain Operation Subclass Behavior Will Be "Preferred" Over Their Parret Class Car Example "Make: strip "mixes deadle" "mix		-
A A Structural Concept That Supports The "IS-A" Relationship An Additive Models Death Spear Yoursel' Levers Of An Otion Parent Class - Sub Class Inherited Behavior Subclasses Can "Overdel" The Behavior Of Their Parent Class by Providing A Officent Implementation of A Certain Operation Subclasses Can "Overdel" The Behavior Of Their Parent Class by Providing A Officent Implementation of A Certain Operation Subclasses Behavior Will Be "Preferred" Over Their Parent Class Car Example Gr —India: "In our "Inherited Support Their Parent Class by Providing A Officent Implementation of A Certain Operation - Subclass Behavior Will Be "Preferred" Over Their Parent Class Car Example Gr —India: "In our "Inherited Support Their Parent Class by Previous Behavior Will Be "Preferred" Over Their Parent Class Car Example Gr —India: "In our "Inherited Support Their Parent Class by Previous Behavior Will Be "Preferred" Over Their Parent Class Car Example Gr —India: "In our "Inherited Support Their Parent Class by Previous Behavior Will Be "Preferred" Over Their Parent Class Car Example Gr —India: "In our "Inherited Support Their Parent Class by Previous Behavior Will Be "Preferred" Over Their Parent Class by Previous Behavior Will Be "Preferred" Over Their Parent Class by Previous Behavior Will Be "Preferred" Over Their Parent Class by Previous Behavior Will Be "Preferred" Over Their Parent Class by Previous Behavior Will Be "Preferred" Over Their Parent Class by Previous Behavior Will Be "Preferred" Over Their Parent Class by Previous Behavior Will Be "Preferred" Over Their Parent Class by Previous Behavior Will Be "Preferred" Over Their Parent Class by Previous Behavior Will Behavior		
- An Additive Mode's - Den't spear strained - LargerOfin Chibo - Parent Class Sub Class Inherited Behavior - Subclass Can "Quentid" The Rehavior of Their Parent Class by Providing A Officent Implementation of A Certain Operation - Subclass Behavior Will Be "Preferred" Over Their Parent Class Subclass Behavior Will Be "Preferred" Over Their Parent Class Car Example [art	Inheritance	
Don't Repeat Yourself Layer of An Chick Parent Class-Sub Class Inherited Behavior Subclasses Can "Override" The Behavior of Their Parent Class By Providing A Different Implementation of A Certain Operation Subclass Behavior Will Be "Preferred" Over their Parent Class Car Example Cartain Subclass Behavior Will Be "Preferred" Over their Parent Class Cartain Subclass Behavior Will Be "Preferred" Over their Parent Class Cartain Subclass Behavior Will Be "Preferred" Over their Parent Class Cartain Subclass Behavior Will Be "Preferred" Over their Parent Class Cartain Subclass Behavior Will Be "Preferred" Over their Parent Class Cartain Subclass Behavior Will Be "Preferred" Over their Parent Class Cartain Subclass Behavior Will Be "Preferred" Over their Parent Class Cartain Subclass Behavior Will Be "Preferred" Over their Parent Class Cartain Subclass Behavior Will Be "Preferred" Over their Parent Class By Providing A Different Implementation of A Certain Operation Cartain Subclass Behavior Will Be "Preferred" Over their Parent Class By Providing A Different Implementation of A Certain Operation Cartain Subclass Behavior Will Be "Preferred" Over their Parent Class By Providing A Different Implementation of A Certain Operation Cartain Subclass Behavior Will Be "Preferred" Over their Parent Class By Providing A Different Implementation of A Certain Operation Operation of A Certain Operation Operat	A Structural Concept That Supports The "IS-A" Relationship	
- Layer of An Cross - Parent Class-Sub Class Inherited Behavior Inherited Behavior - Subclasse Can "Override" The Behavior of Their Parent Class by Providing A Different Implementation Of A Certain Operation - Subclasse Can "Override" The Behavior of Their Parent Class by Providing A Different Implementation Of A Certain Operation - Subclass Behavior Will Be "Preferror" Over their Parent Class Car Example [ar Car Analysis Car Ca		
Inherited Behavior • Subclasse Can "Override" The Behavior Of Their Present Class By Providing A Different implementation of A Certain Operation • Subclass Behavior Will Be "Preferred" Over Their Parent Class Car Example (an Example (an Exam	Layers Of An Onion	-
Subclasses Can "Override" The Behavior Of Their Parent Class By Providing A Different Implementation Of A Certain Operation Subclass Behavior Will Be "Preferred" Over Their Parent Class Car Example Car	- raient Class-sub Class	
Subclasses Can "Override" The Behavior Of Their Parent Class By Providing A Different Implementation Of A Certain Operation Subclass Behavior Will Be "Preferred" Over Their Parent Class Car Example Car		
Subclasses Can "Override" The Behavior Of Their Parent Class By Providing A Different Implementation Of A Certain Operation Subclass Behavior Will Be "Preferred" Over Their Parent Class Car Example Car		-
Subclasses Can "Override" The Behavior Of Their Parent Class By Providing A Different Implementation Of A Certain Operation Subclass Behavior Will Be "Preferred" Over Their Parent Class Car Example Car		
Subclasses Can "Override" The Behavior Of Their Parent Class By Providing A Different Implementation Of A Certain Operation Subclass Behavior Will Be "Preferred" Over Their Parent Class Car Example Car		
Subclasses Can "Override" The Behavior Of Their Parent Class By Providing A Different Implementation Of A Certain Operation Subclass Behavior Will Be "Preferred" Over Their Parent Class Car Example Car		
Subclasses Can "Override" The Behavior Of Their Parent Class By Providing A Different Implementation Of A Certain Operation Subclass Behavior Will Be "Preferred" Over Their Parent Class Car Example Car		
Subclasses Can "Override" The Behavior Of Their Parent Class By Providing A Different Implementation Of A Certain Operation Subclass Behavior Will Be "Preferred" Over Their Parent Class Car Example Car		
Subclasses Can "Override" The Behavior Of Their Parent Class By Providing A Different Implementation Of A Certain Operation Subclass Behavior Will Be "Preferred" Over Their Parent Class Car Example Car		
Subclasses Can "Override" The Behavior Of Their Parent Class By Providing A Different Implementation Of A Certain Operation Subclass Behavior Will Be "Preferred" Over Their Parent Class Car Example Car		_
Subclasses Can "Override" The Behavior Of Their Parent Class By Providing A Different Implementation Of A Certain Operation Subclass Behavior Will Be "Preferred" Over Their Parent Class Car Example Car		
Car Example Car = -mColor: string - mColor: string - mMake: string - mPrice: double + car(make: string, model: string, egetMake(): string; + getMake() : string; + getRode(): string; + getRode(): string; + getRode(): string; + getRode(): string; + setRode(model: string): void; + se	Inherited Behavior	
Car Example Car - Make: string - mCobr: string - mMode: string - mMode: string - mMode: string - more: double + Car(make:string, code:string, color:string, price:double) + getMake(): string; + getColor(): string; + getPrice(): double; + setMake(m Aise-string): void; + setMake(m Mestring): void;	Subclasses Can "Override" The Behavior Of Their Parent Class By	-
Car Example Car		
Car - mMake: string - mColor: string - mModel: string - mPrice: double + Car(make:string, model:string, color:string, price:double) + gettMake(): string; + getMode(): string; + getColor(): string; + getPrice(): double; + setMake(make:string): void; + setMode(model:string): void; + setColor(color:string): void;		
Car - mMake: string - mColor: string - mModel: string - mPrice: double + Car(make:string, model:string, color:string, price:double) + gettMake(): string; + getMode(): string; + getColor(): string; + getPrice(): double; + setMake(make:string): void; + setMode(model:string): void; + setColor(color:string): void;		
Car - mMake: string - mColor: string - mModel: string - mPrice: double + Car(make:string, model:string, color:string, price:double) + gettMake(): string; + getMode(): string; + getColor(): string; + getPrice(): double; + setMake(make:string): void; + setMode(model:string): void; + setColor(color:string): void;		
Car - mMake: string - mColor: string - mModel: string - mPrice: double + Car(make:string, model:string, color:string, price:double) + gettMake(): string; + getMode(): string; + getColor(): string; + getPrice(): double; + setMake(make:string): void; + setMode(model:string): void; + setColor(color:string): void;		
Car - mMake: string - mColor: string - mModel: string - mPrice: double + Car(make:string, model:string, color:string, price:double) + getMake(): string; + getMode(): string; + getColor(): string; + getPrice(): double; + setMake(make:string): void; + setMode(model:string): void; + setColor(color:string): void;		
Car - mMake: string - mColor: string - mModel: string - mPrice: double + Car(make:string, model:string, color:string, price:double) + getMake(): string; + getMode(): string; + getColor(): string; + getPrice(): double; + setMake(make:string): void; + setMode(model:string): void; + setColor(color:string): void;		
Car - mMake: string - mColor: string - mModel: string - mPrice: double + Car(make:string, model:string, color:string, price:double) + getMake(): string; + getMode(): string; + getColor(): string; + getPrice(): double; + setMake(make:string): void; + setMode(model:string): void; + setColor(color:string): void;		
Car - mMake: string - mColor: string - mModel: string - mPrice: double + Car(make:string, model:string, color:string, price:double) + gettMake(): string; + getMode(): string; + getColor(): string; + getPrice(): double; + setMake(make:string): void; + setMode(model:string): void; + setColor(color:string): void;		
Car - mMake: string - mColor: string - mModel: string - mPrice: double + Car(make:string, model:string, color:string, price:double) + getMake(): string; + getMode(): string; + getColor(): string; + getPrice(): double; + setMake(make:string): void; + setMode(model:string): void; + setColor(color:string): void;		
Car - mMake: string - mColor: string - mModel: string - mPrice: double + Car(make:string, model:string, color:string, price:double) + getMake(): string; + getMode(): string; + getColor(): string; + getPrice(): double; + setMake(make:string): void; + setMode(model:string): void; + setColor(color:string): void;		
Car - mMake: string - mColor: string - mModel: string - mPrice: double + Car(make:string, model:string, color:string, price:double) + gettMake(): string; + getMode(): string; + getColor(): string; + getPrice(): double; + setMake(make:string): void; + setMode(model:string): void; + setColor(color:string): void;		
Car - mMake: string - mColor: string - mModel: string - mPrice: double + Car(make:string, model:string, color:string, price:double) + gettMake(): string; + getMode(): string; + getColor(): string; + getPrice(): double; + setMake(make:string): void; + setMode(model:string): void; + setColor(color:string): void;]
- mMake: string - mColor: string - mModel: string - mPrice: double + Car(make:string, model:string, price:double) + getMake(): string; + getMode(): string; + getPrice(): double; + setMake(make:string): void; + setColor(color:string): void; + setColor(color:string): void;		
+ Car(make:string, model:string, color:string, price:double) + getMake(): string; + getColor(): string; + getPrice(): double; + setMake(make:string): void; + setModel(model:string): void; + setColor(color:string): void;	- mMake : string - mColor : string	
+ getMake(): string; + getColor(): string; + getPrice(): double; + setMake(make:string): void; + setModel(model:string): void; + setColor(color:string): void;		
+ getColor() : string; + getPrice() : double; + setMake(make:string) : void; + setModel(model:string) : void; + setColor(color:string) : void;	+ getMake() : string;	
+ setMake(make:string) : void; + setModel(model:string) : void; + setColor(color:string) : void;	+ getColor(): string;	
+setColor(color:string):void;	+ setMake(make:string) : void;	
	+ setColor(color:string): void;	

+ drive() : void;

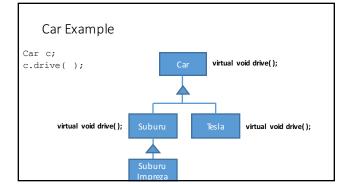


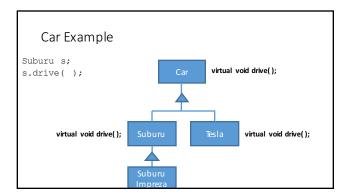


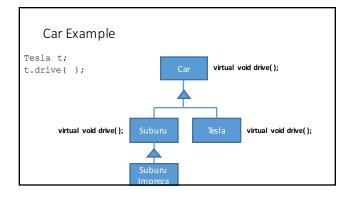


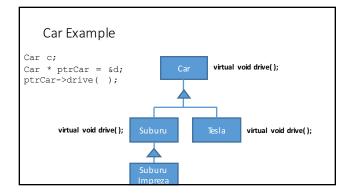
virtual Methods Are Late Binding Methods

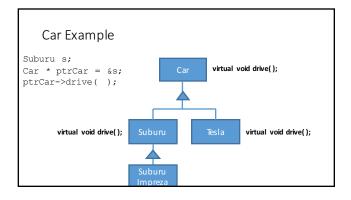
- virtual Marks Methods Whose Behavior Is Expected To Differ In Subclasses
- virtual Methods Are Not Decided At Compile-Time
- \bullet The Compiler Waits Until Run-time To Decide Which Version To Run
 - Yes, C++ Has Run-time Knowledge Of The Structure Of Your Classes

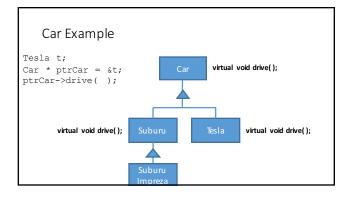


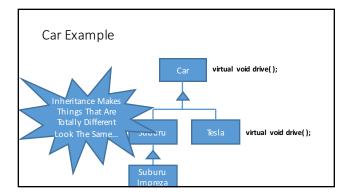












Abstract Classes

- A Class Is Good For Doing Two Things
- 1. Good For Creating Objects Out Of It
- 2. Good For Inheriting From

Abstract Classes

- A Class Is Good For Doing Two Things
- 1. Good For Creating Objects Out Of It
- 2. Good For Inheriting From



Abstract Classes

- "Abstract"
- Quoting From Dictionary.com

Adjective
 1. thought of apart from concrete realities, specific objects, or actual instances:
 an abstract idea.

Abstract Classes

- "Abstract"
- Quoting From Dictionary.com
 - Adjective
 1. thought of apart from concrete realities actual instances:
 an abstract idea.



Car Example Car - mMake: string - mColor: string - mModel: string - mPrice: double + Car(make: string, model: string, color: string, price: double) + getMake(): string; + getModel(): string; + getColor(): string; + getPrice(): double; + setMake(make: string): void; + setModel(model: string): void; + setColor(color: string): void; + setPrice(price: double): void; + setPrice(price: double): void; + drive(): void = 0;

Car Example Car -mMake:string -mColor:string -mModel:string -mPrice:double + Car(make:string, model:string, color:string, price:double) + getMake():string; + getModel():string; + getPrice():double; + setMake(make:string):void; + setVodel(model:string): void; + setColor(color:string): void; + setPrice(price:double): void; + setPrice(price:double): void;

