Yasir Ahmad
Visiting Faculty Member
From
ICS & IT Department
The University Of Agriculture Peshawar

cslearnerr.com

Programming Languages II -- Java

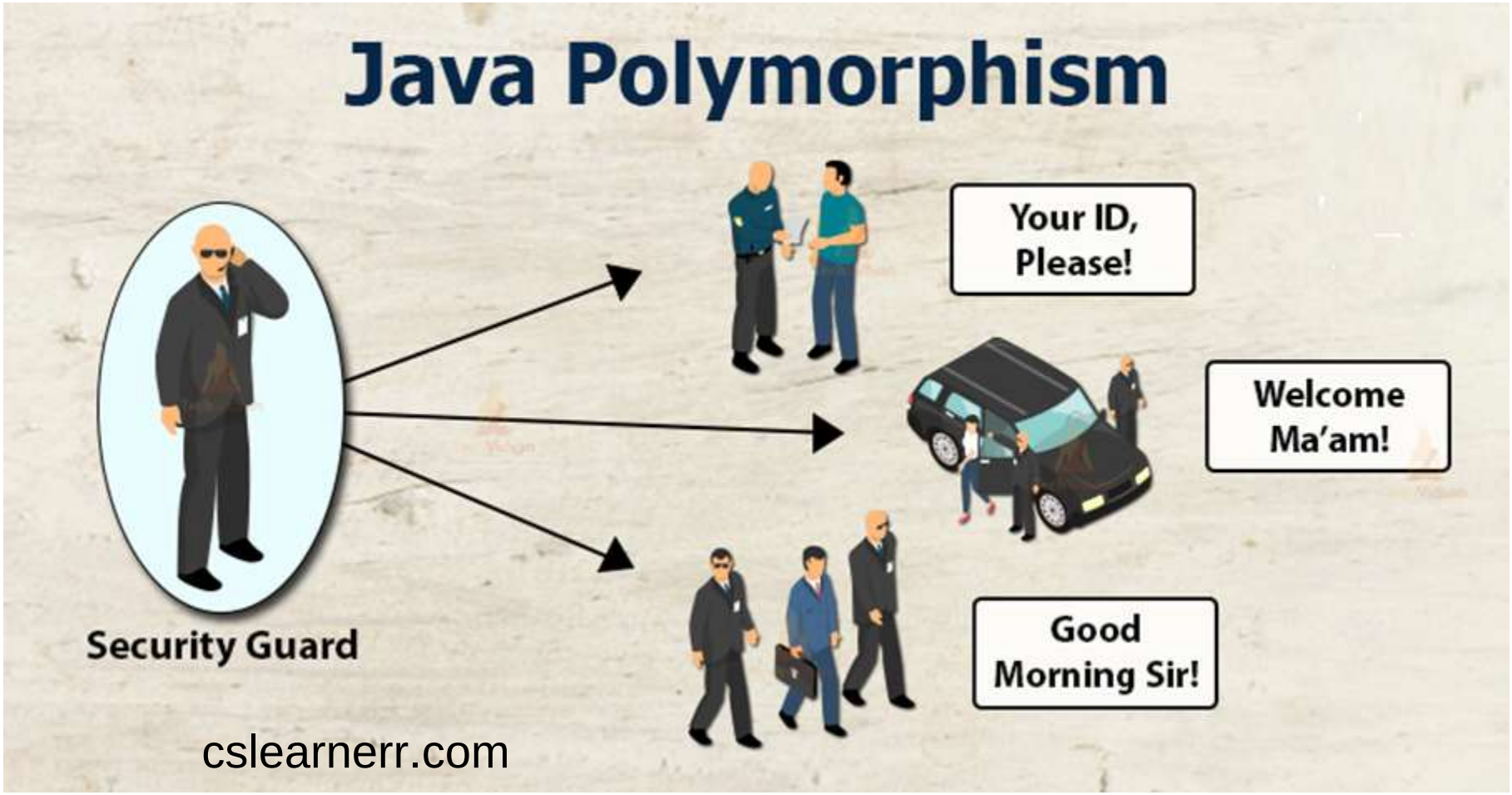
Object Oriented
Programming in **Java-II**

Object Oriented Key Principles

1. Data Abstraction and Encapsulation
2. Inheritance
3. Polymorphism

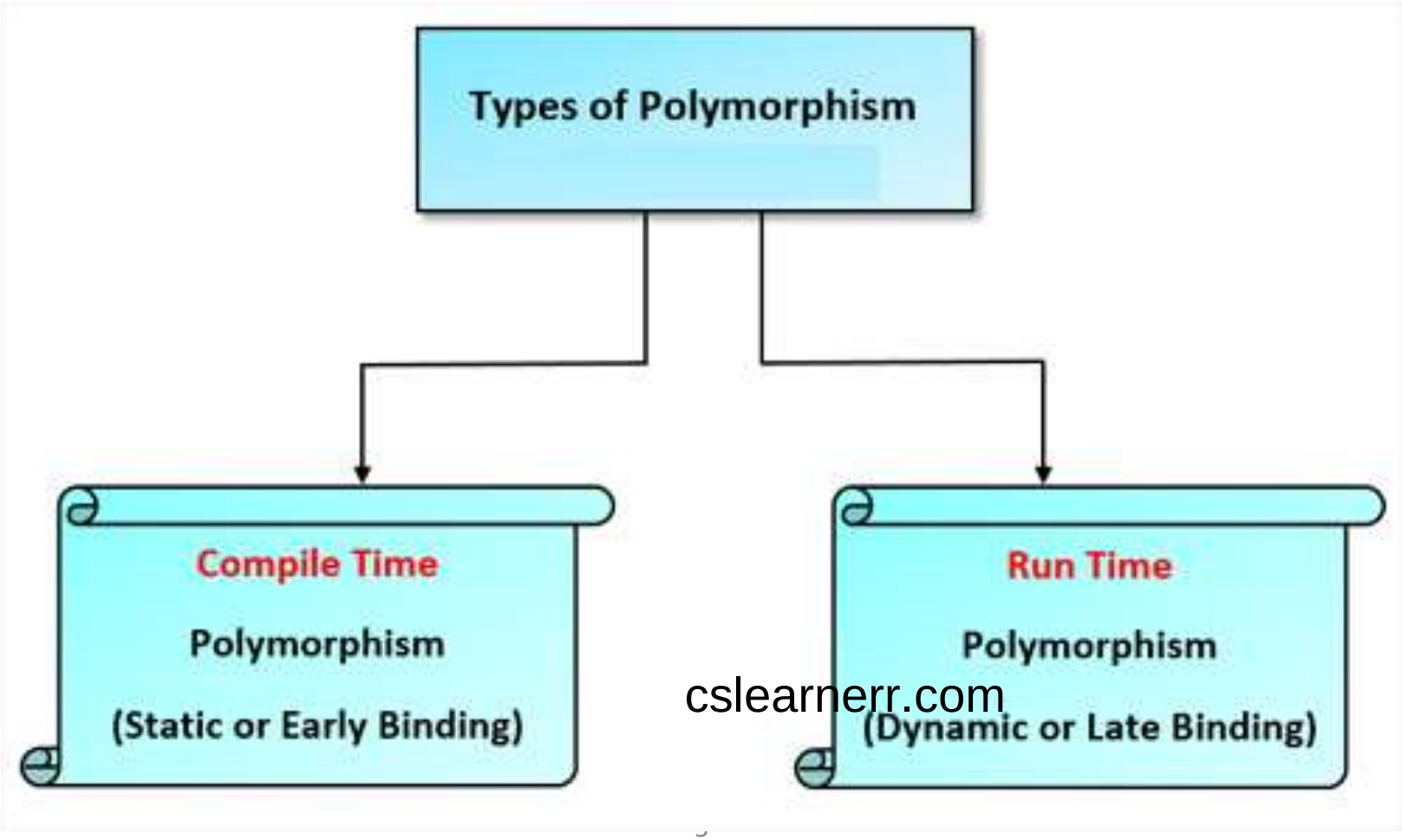
Object Oriented Key Principles

Polymorphism



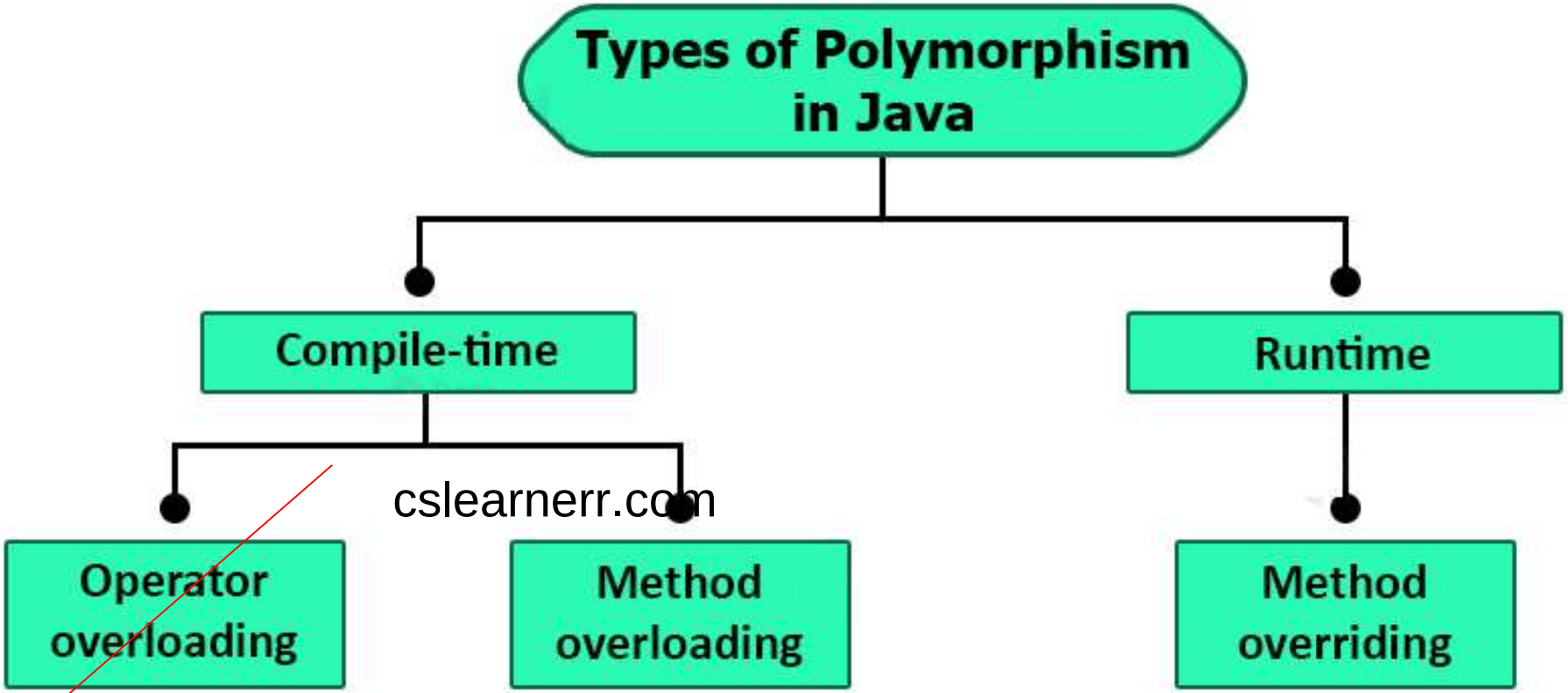
Object Oriented Key Principles

Polymorphism



Object Oriented Key Principles

Polymorphism



cslearnerr.com

or

Constructor overloading

Polymorphism

- Polymorphism - The ability of a superclass variable to behave as a subclass variable
- How?
  - Subclasses inherit all public and protected fields
  - Thus these can be accessed by a superclass variable

Polymorphism

- Limitations!
  - Superclass cannot access new fields and methods within sub classes
  - However if we apply a **cast**, then the cast object reference can access all the new fields and methods

  - int myInt = (int) myDouble; // Manual casting: double to int

Applying polymorphism

- We apply the principle of polymorphism by using the superclass as the generic for an Array List

```
ArrayList<BankAccount> accounts =
new ArrayList<BankAccount>();
```

- Each element in the arraylist can reference a `BankAccount` object
  - or a `BankAccount` subclass object

```
accounts.add(
  new BankAccount("100001", "Mr Shahid"));

accounts.add(
  new CurrentAccount("100003", "Mr Nasir", 100));

accounts.add(
  new SavingsAccount("100005", "Mr Saeed", 5.0));
```

Accessing inherited methods

- When we call a method with multiple versions (overrides)
  - JVM will use the method version which corresponds to the *actual* **object type**
    - Not the reference type
  - I.e. **the overridden version will be used if appropriate**

cslearnerr.com

  - We can simply access the array list element and deal with it as an object of the correct class

Lecture 3, slide 20
Object Instance

- The class, by itself, is a template((سانچے
- Will remain dormant((غیر فعالunless we create an **instance**
- Example
  - `Circle circle = new Circle(10);`
- The left hand side is declaring the object variable

- **The right hand side is instantiating the object by using the class constructor**

Accessing new methods

- When a new (not overridden) method has to be accessed we have to apply a cast
  - Casting to a CurrentAccount object

```
((CurrentAccount)accounts.get(index)).setOverdraftLimit
  (amount);
```

  - Casting to a SavingsAccount object

```
((SavingsAccount)accounts.get(index)).addInterest();
```

- The cast allows the sub class object to overrule the super class reference
- Problem:
  - Applying incorrect sub class will generate an exception!
  - How can we tell which element should be cast?

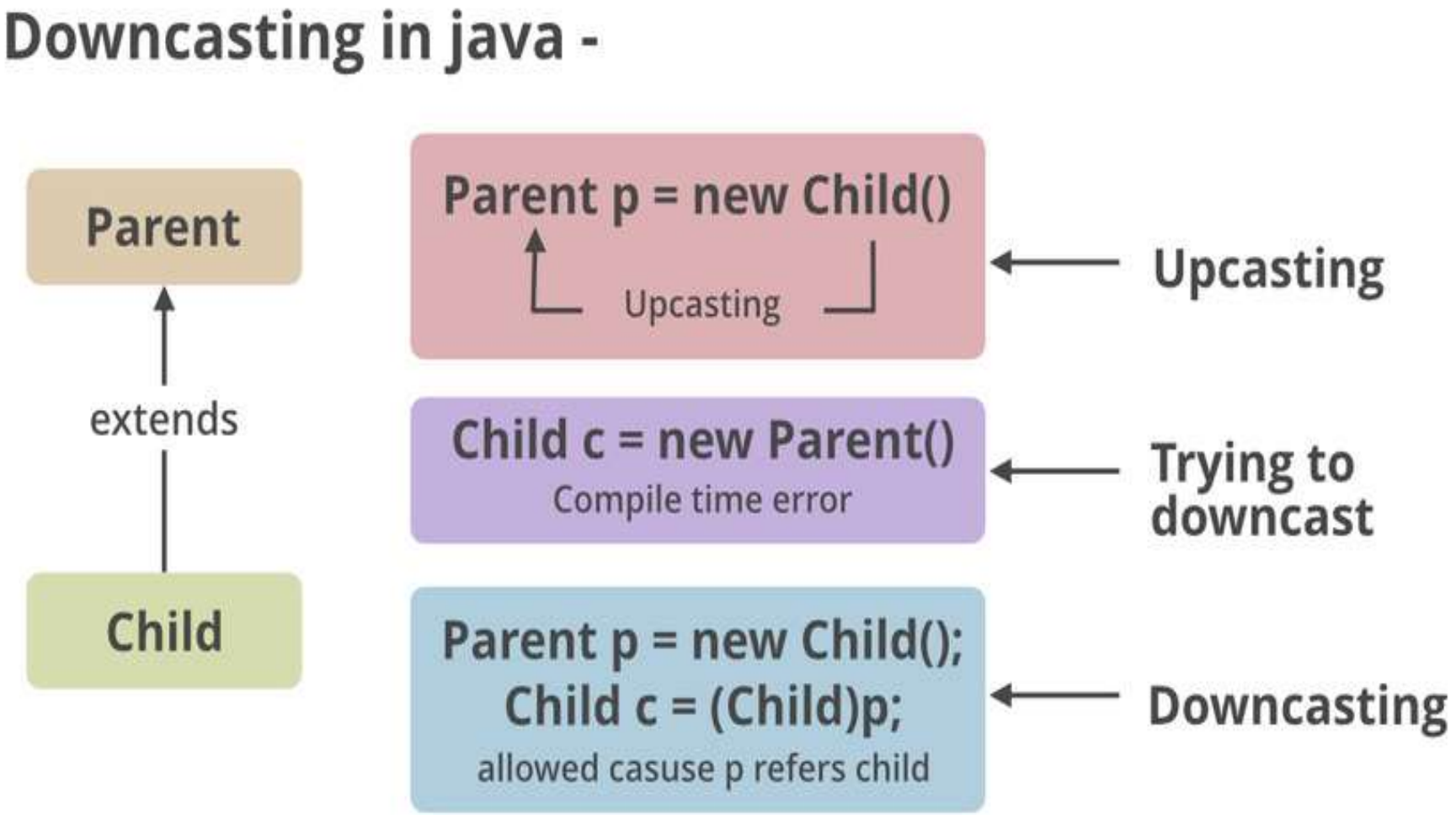
Object Oriented Key Principles

- Data Abstraction and Encapsulation
  - Internal information is hidden and a well defined interface provides access to allowed information
- Inheritance
  - A subclass can *inherit* the fields and methods of a superclass
- Polymorphism
  - A subclass object can be treated as a superclass

Benefits of Using Objects

- Modularity
  - The source code for an object can be written and maintained independently of the source code for other objects. Once created, an object can be easily passed around inside the system.
- Information-hiding
  - By interacting only with an object's methods, the details of its internal implementation remain hidden from the outside world.
- Code re-use
  - If an object already exists , you can use that object in your program.
- Pluggability and debugging ease
  - If a particular object turns out to be problematic, you can simply remove it from your application.

Class objects casting



```
int a = 10;
double b = a; //its automatically casted
System.out.println("b value is "+b);


double c = 2.4;
int d =   (int) c; // its manually casted
System.out.println("d value is "+d);
```

```
BankAccount BA = new BankAccount("100001", "Mr Shahid");                //object is created
CurrentAccount CA = new CurrentAccount("100003", "Mr Nasir", 100);       // object is created
```

```
BankAccount CA2 = new CurrentAccount("100003", "Mr Nasir", 100);          //upcasting

CurrentAccount BA2 =  new BankAccount("100001", "Mr Shahid");             // compile time error

BankAccount CA3 = new CurrentAccount("100003", "Mr Nasir", 100);          //  downcasting
CurrentAccount CA4 = (CurrentAccount) CA3;
```

Assignment (DO IT YOURSELF)

1. Create a super class called Car. The Car class has the following fields and methods.
   a) int speed;
   b) double regularPrice;
   c) String color;
   d) double getSalePrice();

2. Create a sub class of Car class and name it as Truck. The Truck class has the following fields and methods.
   a) int weight;
   b) double getSalePrice(); //If weight>3000, 40%discount.Otherwise,25%discount.

3. Create a subclass of Car class and name it as Sedan. The Sedan class has the following fields and methods.
   a) int length;
   b) double getSalePrice(); //If length>10feet,15%discount,Otherwise,20%discount.

4. Create MainClass class which contains the main() method. Perform the following within the main() method.
   a) Create an instance of Sedan class and initialize all the fields with appropriate values. Use super(...) method in the constructor for initializing the fields of the superclass.
   b) Create two instances of the Truck class and initialize all the fields with appropriate values. Use super(...) method in the constructor for initializing the fields of the super class.
   c) Create an instance of Car class and initialize all the fields with appropriate values.
      Display the sale prices of all instance.