**CASTING IN JAVA(TYPE CASTING)**

**TYPECASTING**:

Converting one type to another type is called type casting

There are two types of casting in java

(i)Primitive casting

(ii)Derived casting

**Q) What is primitive type casting?**

Converting one primitive data type to another primitive data type is called primitive typecasting

Primitive casting can be classified into two types

1. Auto widening

2. Explicit Narrowing

**Auto widening:**

Converting a smaller primitive type to the any of the bigger primitive type is known as widening, since widening is done automatically by the compiler, it’s called as auto widening.

byte short int long float double

**Explicit narrowing:**

Converting the bigger primitive type to the smaller primitive type is known as explicit narrowing.

It should be explicitly specified in the program using ( ) cast operator by the programmer

byte short int long float double

int i=10;;

Double d=10.20;

String s=”Rama”;

**Auto widening:**

Auto widening is done by compiler

Example to understand above point:

int i=10;

long l=i;

Here, long is greater than int. So, when we assign like this compiler will automatically do widening

**Explicit narrowing:**

Explicit narrowing--programmer should explicitly tell to the compiler

Example: to understand above point:

double d1=10.20;

Int k=d1;

We cannot do like this, because int is shorter than byte, compiler will not automatically widen it, we have to mention explicitly,

double d1=10.20;

Int k=(int)d1;

Program:

class Demo64

{

public static void main(String args[])

{

int i=100;

long l=i;// Auto widening

System.out.println(i);

byte b=(byte)l;// Explicit narrowing

System.out.println(b);

}

}

**Q) What is derived casting?**

A) Converting an object to behave like another possible type is called derived casting. In order to convert an object to another type the class should have relationship.

**Derived casting:**

There are two types of derived casting

**Up casting**

**Down casting**

**Up casting:**

Converting an object of sub class to behave like any of the super type. Its automatically done by the compiler hence is called auto up casting.

**Note:**

1.when we upcast, we **can access all members of super class except overridden methods/variables**.

2. After upcasting, subclass cant access all of its own properties,except overridden properties.

3. Upcasting is also known as Generalization and Widening.

Program:

**class** Animal

{

String color;

**void** eat()

{

System.***out***.println("eating");

}

**void** sleep()

{

System.***out***.println("sleeping");

}

**void** sound()

{

System.***out***.println("sound of an animal");

}

}

**class** Dog **extends** Animal

{

//overriden method

**void** sound()

{

System.***out***.println("sound of an dog is bow-bow");

}

**void** bark()

{

System.***out***.println("barking");

}

**void** wagtail()

{

System.***out***.println("wagging tail");

}

}

**class** Demo

{

**public** **static** **void** main(String args[])

{

Dog d1=**new** Dog(); //creation of dog object

d1.sleep();

d1.eat();

d1.bark();

d1.wagtail();

//upcasting Dog to Animal

Animal a1=**new** Dog();// up-casting

//parents members

a1.sleep();

a1.eat();

a1.color="brown";

System.***out***.println("the color og dos is "+ a1.color);

//overriden method

a1.sound();

//after up-casting, it cannot access its own methods.

//a1.bark();

//a1.wagtail();

}

**Down-casting:**

Converting an **up-cast object** to behave like sub class type is called as down-casting

It should be explicitly specified in the program by the user. **Down-casting cannot be directly done**, **only an up-casted object can be down-casted**.

A explicit downcast using ( ) of a super class object to any of the sub class type compile but **throws runtime exception- class cast exception**.

1. When we upcast an object from sub to super class type, this object can access parent class members and except **overridden methods and private and final and static members**

Consider the above program, when we observe

class Demo

{

public static void main(String args[])

{

Dog d1=new Dog(); // normal creation of object

d1.sleep();

d1.eat();

d1.bark();

d1.wagtail();

Animal a1=new Dog();// up-casting

a1.sleep();

a1.eat();

Now, a1 is pointing to Dog object which can access only animal object methods.

We are making sub class to behave like super class

Here, we are creating a sub class and assigning it super class Animal a1 reference variable,

Now we create animal object (which is super class) and assign it to Dog d reference variable(which is sub class)

Dog d=new Animal();

This is called **direct down-casting.**

This will **throw compile time error** -stating that not compatible type error

So, super class will never behave like sub class, Ex: **every dog will behave like an animal, but every animal cannot behave like dog**.

We will see whether above stating is possible or not using casting

Dog d=(Dog)new Animal();

This is called explicit down-casting.

This is also not possible.

This is will compile, but during execution JVM will throw “**class cast Exception**”.

**How to down cast then?**

Observe....

Here,

Animal a1=new Dog();// up-casting

a1.sleep();

a1.eat();

Dog is up-casted, Now we are down-casting the up-casted object

Dog d=(Dog) a1;

we are assigning the a1 to the Dog reference variable d using cast operator.

|  |  |  |
| --- | --- | --- |
| **S.No** | **Upcasting** | **Downcasting** |
| 1. | A child object is typecasted to a parent object. | The reference of the parent class object is passed to the child class. |
| 2. | We can perform Upcasting implicitly or explicitly. | Implicitly Downcasting is not possible. |
| 3. | In the child class, we can access the methods and variables of the parent class. | The methods and variables of both the classes(parent and child) can be accessed. |
| 4. | We can access some specified methods of the child class. | All the methods and variables of both classes can be accessed by performing downcasting. |
| 5. | Parent p = new Child() | Parent p = new Child() Child c = (Child)p; |