# Java vs C++ (difference)

| Java | C++ |
|---|---|
| class kapil<br>{ public static void main(String args[]) throws Exception<br>{ System.out.println("Hari"); }<br>} file name a.java javac a.java java kapil | #include<iostream><br>using namespace std;<br>main( ){ cout<<" Mohan"; }<br>filename t.cpp c++ t.cpp ./a.out |
| class a{ public int u; void g(){b x; x=new b();x.k=2;}}<br>class b{ public int k; void h(){a y; y=new a();y.u=9;}}<br>class kapil { public static void main(String z[])<br>{ System.out.println("hari"); } }<br>Java neede s object instantiation (new) | class a{public: int u; void g( ); };<br>class b{public: int k; void h( ){a y; y.u=9;} };<br>void a::g( ){b x; x.k=2;}<br>main( ) { cout<<" Hari"; }<br>Header may be required in C++ |

Inheritance supports overloading in Java but not in C++

| | |
|---|---|
| class a{ void g( ){SOP("anil");} } | class a{ public: void g( ){cout<<"anil";} }; |
| class b extends a { void g(int w){SOP(10*w);} } | class b: public a { public: void g(int w){cout<<10*w;} }; |
| main: b y; y=new b( ); y.g(2); | main( ){ b y; y.g(2); } |
| y.g( ) is no error. | y.g( ) is error. removal of void g(int w){..} removes error. |

Superclass pointer points at subclass (Java: uses function of subclass o/p hari c++: own class o/p anil

| | |
|---|---|
| class a{ void g( ){SOP("anil");} } | class a{ public: ~~virtual~~ void g( ){cout<<"anil";} }; |
| class b extends a { void g( ){SOP("hari");} } | class b: public a { public: void g( ){cout<<"hari";} }; |
| main: a x; b y; y=new b( ); x=y; x.g( ); | main( ){ a *x; b y; x=&y; (*x).g( ); } |

Implicit pointer in Java (Java o/p 4 x and y are pointers c++:o/p3 x and y are objects)

| | |
|---|---|
| class a{ int p,q; } | class a{ public: int p,q; }; |
| main:a x,y;x=new a( );y=new a( );x.p=3;y=x;y.p=4;SOP(x.p); | main(){a x,y;x.p=3;y=x;y.p=4;cout<<x.p;} |

More than one class can not be inherited in Java (use class c extends a,b is error)

| | |
|---|---|
| class a{ } class b{ } class c extends a{}<br>class kapil{public static void main(String z[]) {c x;}} | class a{ }; class b{ }; class c: public a, public b{};<br>main( ){ c x; } |

Name conflict In c++ class name and function name can not be same.

| | |
|---|---|
| class a{ a( ){SOP("h");} void a(){SOP("a");} }<br>class kapil{public static void main(String z[]){a t=new a();t.a();}} | class a{public:a(){cout<<"h";} void b(){cout<<"a";}};<br>main( ){a t; t.b(); } |
| output is ha. Replacement of a by b in a(){SOP("h");} causes error.<br>In function the return type is void. A constructor has no return type | Replacement of 'b' by 'a' causes error |

Default Visibility In Java default visibility is public. In C++ default visibility is private.

| | |
|---|---|
| class p{int a; private int b; }main:p x;x=new p( ); x.a=2; | class p{ int a; public: int b; };main( ){ p x; x.b=2; } |
| Use of x.b=2; causes error. | Use of x.a=2; causes error. |

class a{ public: int p; ~a( ){ cout<<p*2; } }; int f( ){ a x; x.p=7; cout<<"n"; } //C++ has Destructor
main( ){ a t; f( ); cout<<"y"; { a u; u.p=8; cout<<"z"; } cout<<"w";} o/p n14yz16w(somenumber)

Automatic type conversion

| | |
|---|---|
| class a{ void g(int a){SOP(a+2);}<br>void g(float a){SOP(a+3);}<br>void g(double a){SOP(a+4);}}; | class a{ public: void g(int a){cout<<a+2;}<br>void g(float a){cout<<a+3;}<br>void g(double a){cout<<a+4;}}; |
| main: a x=new a( ); int p=5; float q=5; double r=5; x.g(p); | main( ){a x;int p=5;float q=5; double r=5; x.g(p);} |

S:remove void g(int a) T:remove void g(float a) U:remove void g(double a) ambi:ambigity

| | o/p | S | | T | | U | | ST | | SU | | TU | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Java | C++ | Java | C++ | Java | C++ | Java | C++ | Java | C++ | Java | C++ |
| x.g(p) | 7 | 8 | ambiguity | | | | | 9 | 9 | 8 | 8 | | |
| x.g(q) | 8 | | | 9 | 9 | | | 9 | 9 | | | error | 7 |
| x.g(r) | 9 | | | | | error | ambiguity | | | error | 8 | error | 7 |

C++: An integer has equal preference for conversion into float and double. A double has equal preference for conversion into float and int. But float has first preference for conversion into double and second into int.

Java: A high precision data can not be converted into lower precision data. An double can not be converted into either int or float. A float can not be converted into int. An int has first preference for conversion into float, second into double.

C++ → two options → ambiguity due to equal preference.
Java → no options → error

Method and binary operator (The output of each of the following is 105)

| | | |
|---|---|---|
| class hari { public: int a;};<br>void g(hari k, int p)<br>{ cout<<k.a+p;}<br>main( )<br>{ hari u; u.a=5; g(u,100);} | class hari { public: int a;};<br>void operator -(hari k, int p)<br>{ cout<<k.a+p;}<br>main( )<br>{ hari u; u.a=5; u-100; } | class hari<br>{ public: int a;<br>void g(int p){cout<<a+p;}};<br>main( ){ hari u; u.a=5;<br>u.g(100);} |
| class hari<br>{ public: int a;<br>void operator *(int p)<br>{cout<<a+p;} };<br>main( ){ hari u; u.a=5; u*100;} | class hari { public: int a;};<br>int g(hari k, int p){ return(k.a+p);}<br>main( )<br>{ hari u; int t; u.a=5;<br>t=g(u,100); cout<<t; } | class hari { public: int a;<br>int operator *(int p)<br>{return(a+p);} };<br>main( ) { hari u; int t;<br>u.a=5;t=u*100; cout<<t; } |

Constructor vs Function(method)

class b{ public: int p,q; b(int w){ p=w+7; q=w-5; } void g(int w){ p=w+7; q=w-5; } };
main( ){ b x(3); cout<<x.p<<","<<x.q; x.g(9); cout<<x.p<<","<<x.q; } output 10,-2 16,4
(1) The constructor name is same as class name. (2) No return type (void is not used)

Current vs Parameterized Object

class anil { public: int x; void g(anil m){ cout<<x*100+m.x; } };
main( ) { anil u,v; u.x=5; v.x=3; u.g(v); v.g(u); } The output is 503 and 305.
u.g(v) transforms cout<<x*100+m.x into cout<<u.x*100+v.x; Here `u` is the current object and `v` is a parameterized object. Same holds for operator.
Use void operator +=(anil m){ cout<<x*100+m.x; } and u+=v in place of u.g(v).

Multiple Object

class anil{ public: int x;}; class ravi{public: int y,z;};
void k(anil a,ravi b){int p=a.x+b.y;cout<<p*b.z;}
main(){ anil u;ravi v; u.x=5; v.y=3; v.z=9; k(u,v);} output (5+3)*9=72

Object returned

In above put this function: ravi k(anil a){ravi p; p.y=a.x+7; p.z=a.x*2; return p;}
main(){ anil u; ravi v; u.x=5; v=k(u); cout<<v.y<<","<<v.z; } output12,10
Using operator: ravi operator+(anil a){...} in main v=k(u) is replaced by v=+u;
The above can also be written as following
class ravi{public: int y,z;};
class anil{ public: int x; ravi k( ){ravi p; p.y=x+7; p.z=x*2; return p;} };
main(){ anil u; ravi v; u.x=5; v=u.k(); cout<<v.y<<","<<v.z; } do it using operator

Inheritance

class b{ public: void f(){cout<<"gyan";} void h(){cout<<"hari";} };
class t: public b { public: void h(){cout<<"anil";} };
main() { t x; b y; x.f( ); x.h( ); x.b::h( ); y.h( );} output gyan anil hari hari
x.f() tries to execute function f() from class `t`. Since it is absent hence it takes from its super class `b`. x.h() executes h() of t since it is present. x.b::h() executes `h` of class b since it is mentioned.
Let us put one more class c { public: void f(){cout<<"mohan";} }
Let us use class t: public b, public c Here x.f( ) shows ambiguity. x.c::f() outputs mohan

class b{ public: int p,q; void g( ){ cout<<p*q;} };
class t: public b{ public: int q,r; void f( ){cout<<p*q;} };
main( ){ t x; x.p=5; x.q=6; x.r=7; x.b::q=8; x.f( ); x.g( ); } output 30 40
`x` has 4 components. [q=6,r=7,p=5,super(q)=8] x.f() executes f() of class `t`.
Since `g` is absent hence x.g() executes `g` of class `b`. Hence super(q) is used.

Inheritance fails when overloading

class b{ public: void f(int a){cout<<a+5;} void f(int a, int b){cout<<a*b;} };
class t: public b { public: void g(double a){cout<<a*2;} };
main( ){ t x; x.f(3); x.f(3,5); } The output is 8 15. When `g` is replaced by `f` then x.f(3,5) is error. x.f(3) outputs 3.0*2=6 (using type conversion, not 3+5 using inheritance)

Template

class a{public: void f( ){cout<<"hari";}}; class b{public: void f( ){cout<<"ravi"; }};
void g( ){a x; x.f( );} void h( ){b x; x.f( );} main( ){g( ); h( );}
The output of the above and following program are same as hariravi. m<a>=g m<b>=h.
Class a, b same. template<class k> void m( ){k x; x.f( );} main( ){m<a>( ); m<b>( );}