Compare C++ Syntax and Features with Java

Here is a brief summary that compares C++ syntax and features with Java

C++	Java
cout << "something" << endl;	System.out.println("something");
int x; cin >> x;	Scanner input = new Scanner(System.in); int x = input.nextInt();
main function (not in a class, return int)	main method in a class (void)
namespace	pacakge
using namespace	import a package
short, int, long, unsigned short, unsigned int, unsigned long, float, double, long double, char, bool	byte, short, int, long, float, double, char, boolean
Casting: (int)x or static_cast<int>x	(int)x
operand evaluation order a + b	a + b (strictly from left to right)
||, &&, !	||, &&, !, ^, |, &
if, switch, conditional operator	if, switch, conditional operator
for, while, do-while	for, while, do-while
break, continue	break, continue
function	method
Pass-by-value	Pass-by-value
Pass-by-reference for any type	No pass-by-reference
overloading	overloading
Ambiguous overloading	Ambiguous overloading
Global variables	
Static local variables	
int counts[4]	int[] counts, int counts[] new int[4];
	Arrays are objects
counts = counts1; (not allowed)	counts = counts1; (OK)
Static allocation and heap allocation (new)	
Bound checking (no)	Bound checking (checked at runtime)

Cannot get the length from the array	counts.length returns the length of an array
f(counts, SIZE)	f(counts)
pointers	
C-string	
Circle class	Circle class
Create an object Circle c; // Create an object, c is the name of the object. The object is created using the no-arg constructor.	Circle c; // Declare a reference variable, c is just a reference to the object
Circle c(5.5);	
c.getArea();	
Circle* p = new Circle(); // Dynamic creation	Circle c = new Circle(); // All objects in Java are created dynamically
p->getArea(); or (*p).getArea()	c.getArea();
delete p;	c = null; Automatically removed
Pass-by-value function f(Circle c) Invoke f(c1)	
Pass-by-reference function(Circle& c) Invoke f(c1)	
Pass-by-reference via pointer function(Circle* p) invoke f(&c1)	Pass-by-value in Java method m(Circle c) invoke m(c1)
The string class	The String class
static members (data fields and functions)	static members (data fields and methods)
Header files Define class in a .h file and implement it in a .cpp file	Define class and implement all methods in the same class .java file
Include header file inclusion guard (to avoid multiple inclusion) ifndef then define,	
Default constructor: same	Default constructor: same
Copy constructor	clone() method
Circle c1; Circle c2(5.5); c1 = c2; // Still two different objects	Circle c1 = new Circle(); Circle c2 = new Circle(5.5); c1 = c2; // c1 and c2 refer to the same

	object
Anonymous object Circle().getArea();	new Circle().getArea()
destructor	The Object class has the finalize() method.
dynamic_cast<type*>(pointer)	(type)objectReference
Constructor chaining: same	Constructor chaining: same
Destructor chaining	Invoking finalize() methods from the current class on to the Object class
The functions invoked from a constructor in C++ are not polymorphic, i.e., the functions are statically bound.	The methods invoked from a constructor in Java are polymorphic, i.e., the methods are dynamically bound.
Immutable objects: (memberwise copy)	Immutable objects: String, Integer, String s1; String s2; s1 = s2;
Inheritance (Derived class, base class)	Inheritance (subclass, superclass)
Constructor chaining	Constructor chaining

class A : public B
{
 	
};
	class A extends B
	{
	
	}

private, protected for the members of the class	private, protected for the members of the class
public class A (no such thing in C++)	public class,
Encapsulation, inheritance, and polymorphism	Encapsulation, inheritance, and polymorphism
Redefine functions	No redefine functions and only override functions
Override (virtual function)	override
Abstract class	Abstract class

class A
{
  public:
  virtual void m() = 0;
}
	public abstract class A {
	  public abstract void m();
	}

No interface	interface
Multiple inheritance	No multiple inheritance in Java

class A : public B, public C
{
}
	(wrong)
	public class A extends B, C {
	}

Text I/O ifstream	Text I/O Scanner

ofstream	PrintWriter
Binary I/O istream opened with binary mode reinterpret_cast	Binary I/O InputStream OutputStream FileInputStream FileOutputStream BufferedInputStream BufferedOutputStream ObjectInputStream ObjectOutputStream
Operator overloading s[0] cout << s1 cin >> s1 s1 + s2 s1 < s2	Not in Java

Exception handling
 	
try
{
 	
}
catch (type e)
{
 	
}
finally
{
 	
}
 	
throw an object or primitive type value
	Exception
	
	try
	{
	
	}
	catch (ObjectType e)
	{
	
	}
	finally
	{
	
	}
	
	throw an object of Throwable instance

Templates You can substitute a generic type with a primitive type or object type	Generics You can substitute a generic type with only object type
STL	Collections framework

list, vector, deque set, multiset, map, multimap queue, stack, priority_queue STL algorithms Uses iterators extensively	Collection, List, Set, Map HashSet, LinkedHashSet, TreeSet, LinkedList, ArrayList, HashMap, LinkedHashMap, TreeMap Stack, Queue, Priority_Queue No counterpart of algorithms and algorithms are built in the API

foreach loop in C++11 (Visual C++ 2012)
for (type& e: collection)
{
}
	foreach loop in Java
	for (type e: collection)
	{
	}

auto type	No auto type