ساختمان داده ها

آشنایی با C++ برای برنامه نویسان جاوا

مدرس: غیاثی‌شیرازی
دانشگاه فردوسی مشهد

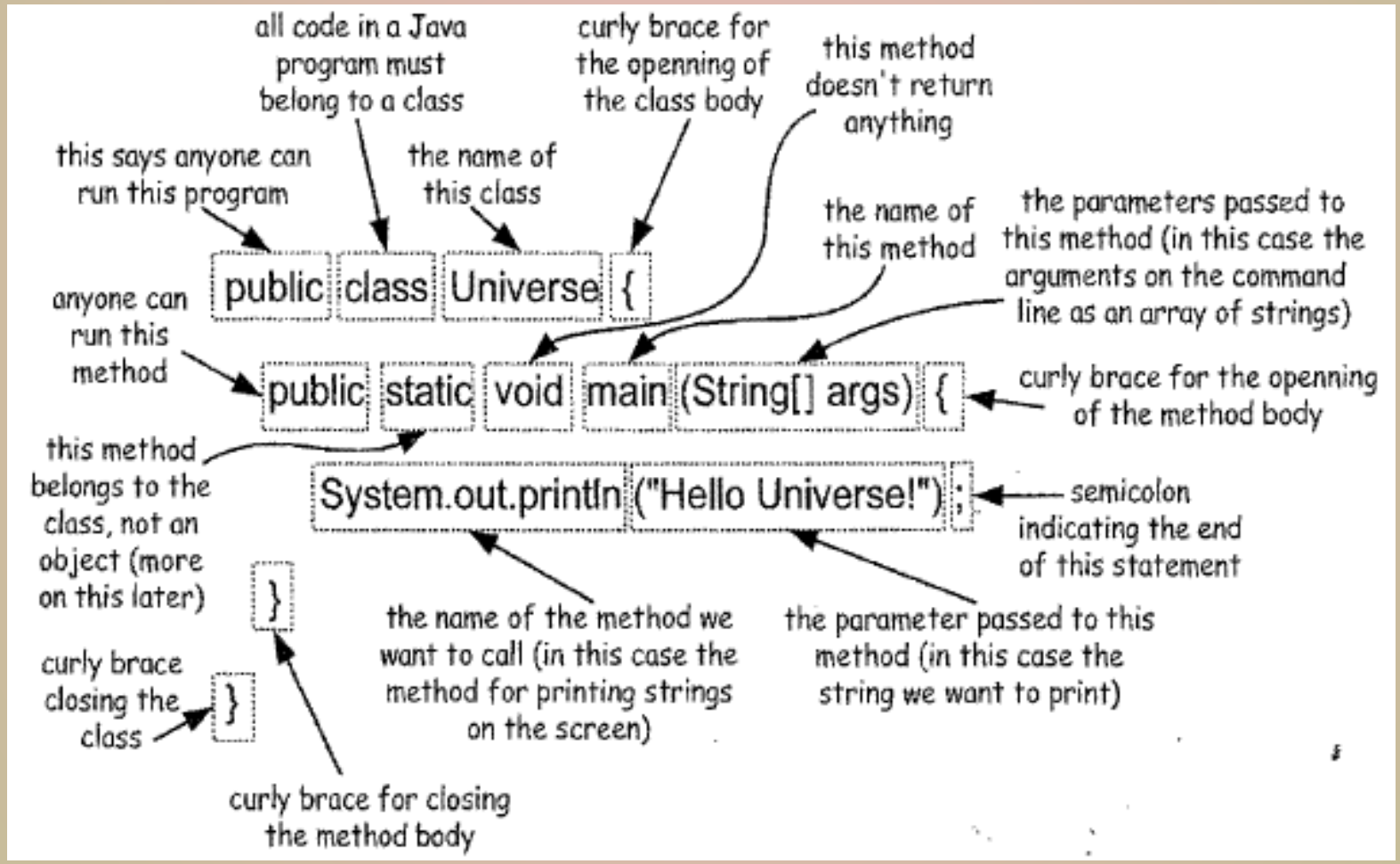
مقدمه

- زبان جاوا بسیاری از قابلیت های زبان C++ را حذف کرده است.
  - سربارگذاری عملگرها (Operator Overloading)
  - ارث بری چندگانه
  - ساخت متغیرها در پشته
  - توابع غیر مجازی
  - آزاد سازی حافظه
  - متود مخرب
- در این ارائه از زبان جاوا شروع می کنیم و معادل آن را در زبان C++ بیان می کنیم.
- بنابراین تمام قابلیت های زبان C++ گفته نمی شود.

کلمات کلیدی مخصوص جاوا

Reserved Words			
abstract	else	interface	switch
boolean	extends	long	synchronized
break	false	native	this
byte	final	new	throw
case	finally	null	throws
catch	float	package	transient
char	for	private	true
class	goto	protected	try
const	if	public	void
continue	implements	return	volatile
default	import	short	while
do	instanceof	static	
double	int	super	

نقطه شروع برنامه در جاوا



نقطه شروع برنامه در C++

```
 1  #include <cstdlib>
 2  #include <iostream>
 3  /*  This program inputs two numbers x and y and outputs their sum */
 4  int main( ) {
 5    int x, y;
 6    std::cout << "Please enter two numbers: ";
 7    std::cin >> x >> y;                          // input x and y
 8    int sum = x + y;                             // compute their sum
 9    std::cout << "Their sum is " << sum << std::endl;
10    return EXIT_SUCCESS;                         // terminate successfully
11  }
```

انواع داده پایه در جاوا

boolean	Boolean value: true or false
char	16-bit Unicode character
byte	8-bit signed two's complement integer
short	16-bit signed two's complement integer
int	32-bit signed two's complement integer
long	64-bit signed two's complement integer
float	32-bit floating-point number (IEEE 754-1985)
double	64-bit floating-point number (IEEE 754-1985)

اشیاء عددی در جاوا

Base Type	*Class Name*	*Creation Example*	*Access Example*
byte	Byte	n = new Byte((byte)34);	n.byteValue()
short	Short	n = new Short((short)100);	n.shortValue()
int	Integer	n = new Integer(1045);	n.intValue()
long	Long	n = new Long(10849L);	n.longValue()
float	Float	n = new Float(3.934F);	n.floatValue()
double	Double	n = new Double(3.934);	n.doubleValue()

انواع داده پایه در C++

bool	Boolean value, either **true** or **false**
char	character
short	short integer
int	integer
long	long integer
float	single-precision floating-point number
double	double-precision floating-point number

- توجه: C++ زبان سیستم است و تعریف داده های فوق (بویژه int و long) بسته به سخت افزاری که برنامه بر روی آن اجرا می شود ممکن است کمی متفاوت باشد.

انواع شمارشی در جاوا

```
public class DayTripper {
  public enum Day {MON, TUE, WED, THU, FRI, SAT, SUN};
  public static void main(String[] args) {
    Day d = Day.MON;
    System.out.println("Initially d is " + d);
    d = Day.WED;
    System.out.println("Then it is " + d);
    Day t = Day.valueOf("WED");
    System.out.println("I say d and t are the same: " + (d == t));
  }
}
```

The output from this program is:

```
Initially d is MON
Then it is WED
I say d and t are the same: true
```

انواع شمارشی در C++

```
enum WeekDay {SAT, SUN, MON, TUE, WED, THU, FRI};
//enum WeekDay {MON=2, TUE, WED, THU, FRI, SAT=0, SUN };

void main ()
{
        WeekDay wd = SAT;
        int i = (int) wd;
        cout << "wd = "<< wd << "\t" << "(int) wd = "<<  i;
        wd = (WeekDay) 1; // wd = SUN;
}
```

Output: wd = 0 (int) wd = 0

تعریف کلاس در جاوا

```
public class Counter {
  protected int count; // a simple integer instance variable
  /** The default constructor for a Counter object */
  Counter() { count = 0; }
  /** An accessor method to get the current count */
  public int getCount() { return count; }
  /** A modifier method for incrementing the count */
  public void incrementCount() { count++; }
  /** A modifier method for decrementing the count */
  public void decrementCount() {count--; }
}
```

سطح دسترسی کلاس

- public
  - این کلاس توسط هر کلاس دیگری قابل مشاهده است.

- خالی (که به آن package-private گفته می شود)
  - این کلاس تنها توسط کلاس های همین بسته قابل مشاهده است.

سطح دسترسی اعضا

- public
  - عضو توسط همه کلاس ها قابل دسترسی است.
- protected
  - عضو توسط بسته جاری و همچنین خود کلاس و هر کلاسی که از آن ارث ببرد قابل دسترسی است.
- package-private
  - تنها توسط کلاس های همین بسته قابل دسترسی است.
- private
  - عضو تنها توسط خود کلاس قابل دسترسی است.

سطوح دسترسی به اعضا در جاوا

Access Levels

Modifier	Class	Package	Subclass	World
`public`	Y	Y	Y	Y
`protected`	Y	Y	Y	N
no modifier	Y	Y	N	N
`private`	Y	N	N	N

تعریف کلاس در C++

```
class Counter {
protected:  int count; // a simple integer instance variable
private:
    /** The default constructor for a Counter object */
    Counter() {count=0;}
    /* In C++, methods cannot have package access. However, it is possible to
    introduce some classes as friends of another class to have access even to
    private attributes and methods. If class Y should be able to create an
    instance of Counter we have:*/
    friend Y;

    public:
    /** An accessor method to get the current count */
    Virtual int getCount() { return count; }
    /** A modifier method for incrementing the count */
    Virtual void incrementCount() { count++; }
    /** A modifier method for decrementing the count */
    Virtual void decrementCount() {count--; }
};
```

سطوح دسترسی در C++

- در C++ همه کلاس ها دارای دستری عمومی (public) هستند.
- در C++ مفهوم package وجود ندارد ولی نزدیک ترین مفهوم به آن namespace است.
- سطوح دسترسی در C++ ارتباطی به namespace ندارد.
- اگر سطح دسترسی برای یک عضو تعریف نشود، معادل private است.

اشیاء در جاوا

```
Counter c = new Counter();
c.method1();
```

اشیاء در C++

- در C++ دو نوع روش ساخت شیئ وجود دارد:
روش ساخت شیء در پشته (مخصوص C++)

```
Counter  c;
c.method1();
```

- روش ساخت شیء در Heap (معادل جاوا)

```
Counter  *c = new Counter();
c->method1();
//delete c;
```

تفاوت های حافظه پشته و Heap

- حافظه پشته از ابتدا به یک برنامه اختصاص می یابد و بسته به نوع کامپایل اندازه آن متفاوت است.
  - اندازه حافظه پشته یک برنامه معمولا بسیار محدود است
- حافظه Heap حافظه ای است که حین اجرا (با دستور new) به برنامه اختصاص می یابد و حداکثر اندازه آن برابر حداکثر حافظه آزاد سیستم است.

معادل متغیر final جاوا در C++

Java:

```
class X{
    final int MAX_HEIGHT = 3
}
```

C++:

```
class X{
    const int MAX_HEIGHT = 3
};
```

const در C++

- Read it backwards...
  - int* - pointer to int
  - int const * - pointer to const int
  - int * const - const pointer to int
  - int const * const - const pointer to const int
- the first const can be on either side
  - const int * ↔ int const *
  - const int * const ↔ int const * const

معادل متود final جاوا در C++

- در جاوا هرگاه کلاس پدر بخواهد جلوی سربارگذاری یکی از متودهای خود توسط کلاس های فرزند را بگیرد، آن متود را final تعریف می کند.
- در C++ ننوشتن کلمه virtual در تعریف متود همین تاثیر را دارد.

کلاس های abstract در جاوا

```
abstract class X
{
        public abstract int f();
        public int g(){…};
}
```

کلاس های abstract در C++

```
class X
{
        public: virtual int f()=0;
        public: virtual int g(){…};
};
```

interface

- در جاوا چون ارث بری چندگانه برای کلاس ها ممکن نیست ولی پیاده سازی چندین interface ممکن است، هر دو نوع interface و class وجود دارند.
- در C++ چون ارث بری چندگانه برای کلاس ها هم ممکن است، همه چیز کلاس است.
- بنابراین به جای interface در C++ از همان class استفاده می کنیم.

ارسال پارامتر در جاوا

- در جاوا همه پارامترها به صورت by value فرستاده می شوند.
- البته چون جاوا اجازه دسترسی به خود شیء را نمی دهد، وقتی ارجاع به یک شیء را ارسال می کنیم، ارجاع کپی می‌شود.
- اما چون شیء اصلی کپی نشده است، متود با همان شیء کار می کند. (به همین دلیل ممکن است کسی به اشتباه تصور کند که ارسال پارامترها در جاوا by reference است)

ارسال پارامتر در C++

- در زبان C نیز تنها روش ارسال پارامتر به صورت by value بود.
- در C++ ارسال پارامتر به صورت by reference نیز وجود دارد.
  - اگر قبل از نام پارامتر علامت & قرار گیرد به این معنی است که آن پارامتر به صورت by reference ارسال می شود.

مخرب (Destructor)

- در جاوا مسئولیت آزادسازی حافظه با jvm است.
- در C++ هر شیئ که توسط برنامه نویس new می شود باید توسط خود او delete شود تا حافظه اختصاص یافته به آن آزاد شود.
- به همین دلیل در C++ هر کلاس علاوه بر متود سازنده می تواند دارای متود مخرب نیز باشد که در آن معمولا عملیات آزاد سازی حافظه انجام می شود.
- متود مخرب غیر از آزاد سازی حافظه کاربردهای دیگری نیز دارد.

(Destructor) مخرب

```
Class X  {
public:
        X(int size)  {
                mData = new int[size];
                mSize = size;
        }
        ~X () {
                delete[] mData;
        }
private:
        int*    mData;
        int     mSize;
};
```

رشته ها در جاوا و C++

- نوع char در جاوا ۲ بایتی است و یک حرف Unicode را نشان می دهد در صورتی که در C++ یک بایتی است و یک حرف ASCII را نشان می دهد.
- در جاوا برای نمایش رشته ها از کلاس String استفاده می شود.
- زبان جاوا رشته ها را ("") یک شیء از نوع String در نظر می گیرد.
- زبان C++ رشته ها را ("") به صورت آرایه ای از char در نظر می گیرد.
- در C++ معمولا از کلاس string برای کار با رشته های متنی استفاده می شود که در کتابخانه string تعریف شده است.
- استاندارد C++ ۲۰۱۱، انواع u16string و u32string را نیز در کتابخانه string پیش بینی کرده است.

سوال

- خروجی دستور زیر در C++ چیست؟

```
cout << "Salam"+1 << endl;
```

آرایه ها در جاوا

```
int[] a;
a = new int[20];
for (int i=0;i<a.length;i++)
        a[i]=i;
```

آرایه ها در C++

- در C++ طول آرایه ذخیره نمی شود.

```
int *a;
a = new int[20];
for (int i=0;i<20;i++)
        a[i]=i;
```

شبه دستور #include در C++

- شبه دستور #include مشابه دستور import در جاوا است.
- اگر فایلی که #include می شود در کتابخانه ها باشد:
- #include <string>

- اگر فایلی که #include می شود در برنامه خودمان باشد:
- #include "myheader.h"

ارث بری در جاوا و C++

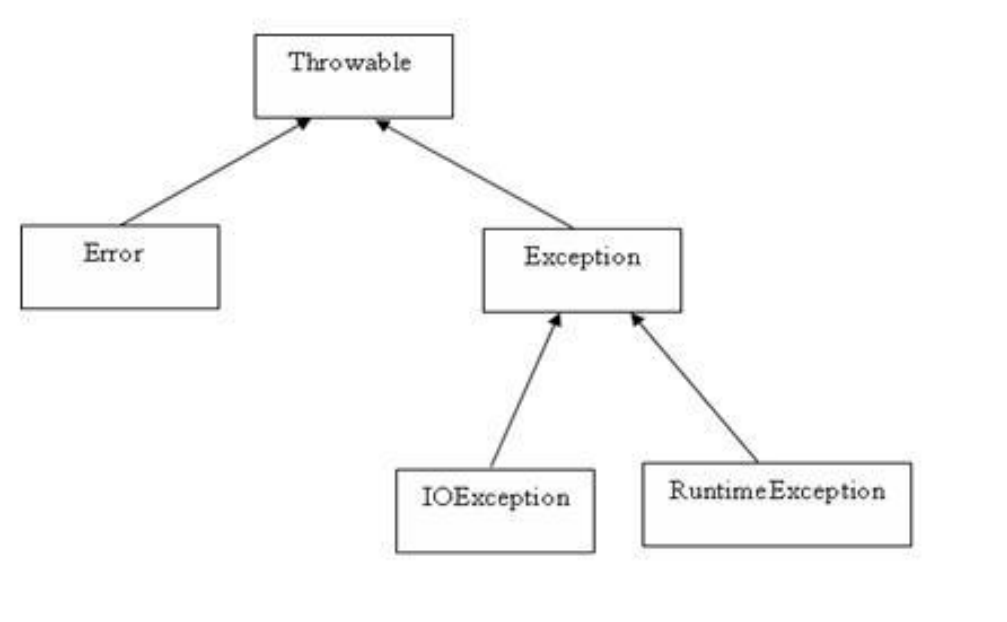
- ارث بری در جاوا

- class D extends B
- class D implements $I_1,\ldots,I_n$
- class D extends B implements $I_1,\ldots,I_n$

- ارث بری در C++

- class D : public $B_1,\ldots,B_n$

انواع Exception در جاوا



- همه exception هایی که توسط یک متود ممکن است پرتاب شوند لازم است در تعریف متود با عبارت throws مشخص شوند (بجز RuntimeException ها).

```
try  {
        method(number);
      }
      catch (IndexOutOfBoundsException ioobe)    {
        System.out.print(ioobe.toString()+'\t');
        return;
      }
      finally   {
        System.out.print ("finally\t");
      }
      System.out.print("end of clause\t");
   }
   public static void method (int i) throws IndexOutOfBoundsException    {
           if (i < 0)
                   throw new IndexOutOfBoundsException();
   }
```

خروجی برای number = -1:

java.lang.IndexOutOfBoundsException finally

خروجی برای number = 1:

finally end of clause

انواع Exception در C++

- در C++ هر نوع شیء ای را می توان throw کرد.

```
class IndexOutOfBoundsException{};
void method (int i) throw (IndexOutOfBoundsException)    {
            if (i < 0)
                        throw new IndexOutOfBoundsException();
}
void main () {
      try{
         method(number);
      }
      catch (IndexOutOfBoundsException* ioobe)    {
            cout << "IndexOutOfBoundsException\t";
            delete ioobe;
      }
      cout << "end of clause\t";
}
```

- خروجی برای number = 1:

end of clause

- خروجی برای number = -1:

IndexOutOfBoundsException end of clause

Widening Casting در جاوا و C++

اگر کلاس D از کلاس B ارث برده باشد، آنگاه هم در C++ و هم در جاوا می توان به جای نمونه ای از B نمونه ای از D را قرار داد.

java

```
D d = new D();
B b = d;
```

C++

```
D* d = new D();
B* b = d;
```

Narrowing Casting در جاوا

- قرار دادن ارجاع b از نوع B در ارجاع d از نوع D باریک کننده است اگر:
- اگر کلاس D از کلاس B ارث برده باشد.
- اگر کلاس D رابط B را پیاده سازی کرده باشد.
- اگر رابط D رابط B را پیاده سازی کرده باشد.

```
B b = new D();
D d;
d = (D) b;
```

اگر شیئ ای که b به آن ارجاع می کند از نوع D نباشد، یک ClassCastException پرتاب می شود.

instanceof در جاوا

```
Number n;
Integer i;
n = new Integer(3);
i = (Integer) n;        // This is legal
n = new Double(3.1415);
i = (Integer) n;        // This is illegal!
```

```
Number n;
Integer i;
n = new Integer(3);
if (n instanceof Integer)
  i = (Integer) n;      // This is legal
n = new Double(3.1415);
if (n instanceof Integer)
  i = (Integer) n;      // This will not be attempted
```

Narrowing Casting در C++

- در C++ اگر اشاره گر b از نوع D نباشد، دستور

```
D* d= (D*)b;
```

خطایی نمی دهد اما برنامه درست عمل نخواهد کرد و در نهایت سیستم عامل جلوی ادامه اجرای آن را می گیرد.

- اما اگر از dynamic_cast استفاده شود، در صورتی که b از نوع D نباشد، مقدار اشاره گر خروجی برابر 0 (NULL) خواهد بود. بنابراین dynamic_cast به نوعی دستور معادل instanceof جاوا در C++ است.

```
D* d = dynamic_cast<D*> b;
If (!d)
        cerr << “b is not of type D*";
```

Template در C++

- در زبان C++ این امکان دیده شده است که یک کلاس به صورت الگو تعریف شود.
- الگوی یک کلاس چند پارامتر مجهول دارد که هر کدام می تواند نام یک کلاس باشد.
- با مشخص شدن پارامترهای مجهول در کلاس الگو، یک کلاس معمولی به دست می آید.
- پارامترهای مجهول علاوه بر نوع ها می توانند موارد دیگری نیز (مانند یک عدد) باشند.

http://www.cplusplus.com/doc/tutorial/templates/

```
template <class T>
class mypair {
    T a, b;
  public:
    mypair (T first, T second)      {a=first; b=second;}
    T getmax ();
};

template <class T>
T mypair<T>::getmax () {
  T retval;
  retval = a>b? a : b;
  return retval;
}

void main () {
  mypair<int>* myobject = new mypair<int>(100, 75);
  cout << myobject->getmax();
}
```

راه حل اولیه جاوا: استفاده از انواع عمومی به جای کلاس های نامشخص

- از آنجا که در جاوا همه اشیاء نهایتا از کلاس Object ارث می برند، یک راه این است که به جای انواع نامشخص از نوع Object و یا یک نوع که پدر همه انواع مجهول است استفاده شود.
- در این صورت لازم است که برنامه نویس نهایتا اشیاء را از نوع عام تر به نوع خاص تر قالب ریزی کند.
- این قالب ریزی اشیاء به نوع خاص باعث شد که از نسخه 5.0 جاوا کلاس های Generics به زبان جاوا اضافه شود.

تعریف انواع Generic در جاوا

```
public class Pair<K, V> {
    K key;
    V value;
    public void set( K k, V v) {
        key = k;
        value = v;
    }
    public K getKey() { return key; }
    public V getValue() { return value; }
    public String toString(){
        return " [" + getKey() + ", " + getValue() + "] ";
    }
}
```

استفاده از انواع Generic در جاوا

```
public static void main (String[] args) {
  Pair<String,Integer> pair1= new Pair<String,Integer>();
  pair1.set(new String("height"), new Integer(36));
  System.out.println(pair1);
}
```

خلاصه

- در این ارائه با مبنا قرار دادن روش برنامه نویسی در جاوا، نحوه تولید کد معادل به زبان C++ شرح داده شد.
- به طور طبیعی، قابلیت هایی از زبان C++ که معادلی در جاوا ندارند شرح داده نشدند (مانند سربارگذاری عملگرها، ارجاع (در مقابل اشاره گر)، ارث بری چند گانه و ...)
- انتظار می رود دانشجو توانایی تولید کد به زبان C++ را داشته باشد ولی الزاما نمی تواند کدهای تولید شده به این زبان توسط افراد دیگر را بخواند.

مطالعه بیشتر

- فصل های ۱ و ۲ مراجع زیر:

- M. T. Goodrich, R. Tamassia, D. M. Mount, *Data Structures and Algorithms in C++, 2nd Edition, 2011.*
- M. T. Goodrich, R. Tamassia, *Data Structures and Algorithms in Java, 5th Edition, 2010.*