Application Program Development

Segment : Major Differences of Java and C++

Outcomes

- Major Differences between C++ and Java

Java and C++

- Both C++ and Java are similar programming languages in terms of various features.
- These languages are helpful for the programming of various apps, operating systems, browsers, and websites.
- Learning these programming languages is quite simple.
- Moreover, the complexity of learning Java and C++ also have a similar level.
- We will discuss some differences between the languages now

Class Definitions

C++

```
class CRectangle {
  int width, height;
public:
  CRectangle (int, int);
  int area () {
  return (width*height);
 }
}; // Note: ";" required here
CRectangle::CRectangle (int a,
                              int b) {
      width = a;
      height = b;
}
```

Java

```
public class CRectangle {
   private int width, height;
   public CRectangle (int a, int b) {
            width = a;
            height = b;
  }
  public int area () {
      return (width*height);
  }
} // Note: no ";" required here
```

Class Definitions

C++

```
class CRectangle {
  int width, height;
public:
  CRectangle (int a, int b);
  int area () {
  return (width*height);
 }
  ~Crectangle(){};
};
CRectangle::CRectangle (int a,
                            int b) {
       width = a;
       height = b;
}
```

Java

```
public class CRectangle {
   private int width, height;
   public CRectangle (int a, int b) {
            width = a;
            height = b;
  }
  public int area () {
      return (width*height);
  }
}
```

- Java does not have destructors.

Class Definitions

C++

```
class CRectangle {
  int width, height;
public:
  CRectangle (int, int);
  int area () {
  return (width*height);
 }
};
CRectangle::CRectangle (int a,
                        int b) {
      width = a;
      height = b;
}
```

Java

```
public class CRectangle {
   private int width, height;
   public CRectangle (int a, int b) {
            width = a;
            height = b;
  }
  public int area () {
       return (width*height);
  }
}
```

Class Definitions

C++ main

```
int main () {
  CRectangle rectA (3,4);
  CRectangle rectB (5,6);
  cout << "rectA area: "
          << rectA.area() << endl;
  cout << "rectB area: "
          << rectB.area() << endl;
  return 0;
}
```

Java main

```
public class CRectangle {
   ...
  //main is inside the class
 //main's type is different
  public static void main(String[] args)
{
  CRectangle rectA = new CRectangle (3,4);
  CRectangle rectB = new CRectangle (5,6);
  System.out.println("rectA area: " +
                                rectA.area());
  System.out.println("rectB area: " +
                                rectB.area());
 }
}
```

Class Definitions Inheritance and Polymorphism

C++

```
#include <iostream>
using namespace std;
class CPolygon {
protected:
  int width, height;
public:
  CPolygon(int a, int b)
  { width=a; height=b;}
  virtual int area() {return 0};
};
class CRectangle: public CPolygon {
 public:
  CRectangle(int a, int b):
  CPolygon(a,b) {}
  int area () {
  return (width * height);
  }
};
```

Java

- **In file Polygon.java:**

```
public class Polygon {
   protected int width, height;
   public Polygon(int a, int b)
      { width=a; height=b;}
   public int area() {return 0};
}
```

- **In file Rectangle.java:**

```
public class Rectangle extends Polygon{
   public Rectangle(int a, int b) {
       super(a,b);
   }
   public int area () {
       return (width * height);
   }
}
```

Class Definitions Inheritance and Polymorphism

C++

```
#include <iostream>
using namespace std;
class CPolygon {
protected:
  int width, height;
public:
  CPolygon(int a, int b)
  { width=a; height=b;}
  virtual int area() {return 0};
};
class CRectangle: public CPolygon {
 public:
  CRectangle(int a, int b):
         CPolygon(a,b) {}
  int area () {
  return (width * height);
  }
};
```

Java

- **In file Polygon.java:**

```
public class Polygon {
   protected int width, height;
   public Polygon(int a, int b)
       { width=a; height=b;}
   public int area() {return 0};
}
```

- **In file Rectangle.java:**

```
public class Rectangle extends Polygon{
   public Rectangle(int a, int b) {
       super(a,b);
   }
   public int area () {
       return (width * height);
  }
}
```

Class Definitions Inheritance and Polymorphism

C++

```
#include <iostream>
using namespace std;
class CPolygon {
protected:
  int width, height;
public:
  CPolygon(int a, int b)
  { width=a; height=b;}
  virtual int area() {return 0};
};
class CRectangle: public CPolygon {
 public:
  CRectangle(int a, int b):
        CPolygon(a,b) {}
  int area () {
    return (width * height);
  }
};
```

Java

- **In file Polygon.java:**

```
public class Polygon {
   protected int width, height;
   public Polygon(int a, int b)
      { width=a; height=b;}
   public int area() {return 0};
}
```

- **In file Rectangle.java:**

```
public class Rectangle extends Polygon{
   public Rectangle(int a, int b) {
       super(a,b);
   }
   public int area () {
       return (width * height);
   }
}
```

Class Definitions Inheritance and Polymorphism

C++

```
class CTriangle: public CPolygon {
  public:
    CTriangle(int a, int b):
         CPolygon(a,b) {}
    int area ()
    { return (width * height / 2); }
};
int main () {
  CRectangle rect(4,5);
  CTriangle trgl(4,5);
  cout << rect.area() << endl;
  cout << trgl.area() << endl;
  return 0;
}
```

Java

- **In file Triangle.java:**

```
public class Triangle extends Polygon {
        public Triangle(int a, int b) {
                super(a,b);
        }
        public int area () {
                return (width * height / 2);
        }
}
```

- **In file PolyTest.java:**

```
public class PolyTest {
        public static void int main () {
                Rectangle rect = new Rectangle(4,5);
                Triangle trgl = new Triangle(4,5);
                System.out.println(rect.area());
                System.out.println(trgl.area());
        }
}
```

Question Time…

How does java delete the DMA?

- Destructor
- Delete Keyword
- Remove Keyword
- [✓ Garbage Collected]

Class Definitions Inheritance and Polymorphism

C++

```
class CTriangle: public CPolygon {
  public:
    CTriangle(int a, int b):
         CPolygon(a,b) {}
    int area ()
    { return (width * height / 2); }
};
int main () {
  CRectangle rect(4,5);
  CTriangle trgl(4,5);
  cout << rect.area() << endl;
  cout << trgl.area() << endl;
  return 0;
}
```

Java

- **In file Triangle.java:**

```
public class Triangle extends Polygon {
        public Triangle(int a, int b) {
                super(a,b);
        }
        public int area () {
                return (width * height / 2);
        }
}
```

- **In file PolyTest.java:**

```
public class PolyTest {
        public static void int main () {
                Rectangle rect = new Rectangle(4,5);
                Triangle trgl = new Triangle(4,5);
                System.out.println(rect.area());
                System.out.println(trgl.area());
        }
}
```

Class Definitions Inheritance and Polymorphism

C++

```
class CTriangle: public CPolygon {
  public:
    CTriangle(int a, int b):
         CPolygon(a,b) {}
    int area ()
    { return (width * height / 2); }
};
int main () {
 CPolygon* rect= CRectangle rect(4,5);
 CPolygon* trgl= CTriangle trgl(4,5);
  cout << rect.area() << endl;
  cout << trgl.area() << endl;
  return 0;
}
```

Java

- **In file Triangle.java:**

```
public class Triangle extends Polygon {
        public Triangle(int a, int b) {
                super(a,b);
        }
        public int area () {
                return (width * height / 2);
        }
}
```

- **In file PolyTest.java:**

```
public class PolyTest {
        public static void int main () {
                Polygon rect = new Rectangle(4,5);
                Polygon trgl = new Triangle(4,5);
                System.out.println(rect.area());
                System.out.println(trgl.area());
        }
}
```

Pure Virtual VS Abstract classes

C++

```
#include <iostream>
using namespace std;
class CPolygon {
protected:
  int width, height;
public:
  CPolygon(int a, int b)
  { width=a; height=b;}
  virtual int area() = 0;
};
 ...
int main () {
 CPolygon *rect = new CRectangle(4,5);
 CPolygon *trgl = new CTriangle(4,5);
// illegal
//CPolygon *poly = new CPolygon(4,5);
 cout << rect->area() << endl;
 cout << trgl->area() << endl;
 //cout << poly->area() << endl;
 return 0;
}
```

Java

- **In file Polygon.java:**

```
public abstract class Polygon {
   protected int width, height;
   public Polygon(int a, int b)
            { width=a; height=b;}
    public abstract int area();
}
```

…

- **In file PolyTest.java:**

```
public class PolyTest {
    public static void int main () {
    Polygon rect = new Rectangle(4,5);
    Polygon trgl = new Triangle(4,5);
    // illegal
    // Polygon poly = new Polygon(4,5);
    System.out.println(rect.area());
    System.out.println(trgl.area());
    // System.out.println(poly.area());
}
}
```

→Static Keyword←

- **Static Block:**
  - Unlike C++, Java supports a special block, called static block (also called static clause) which can be used for static initialization of a class. This code inside the static block is executed only once.
- **Static Local Variable:**
  - Unlike Java, C++ supports static local variables.

Exceptional Handling

C++

```
#include <iostream>
using namespace std;
class CPolygon {
protected:
  int width, height;
public:
  CPolygon(int a, int b)
  { width=a; height=b;}
  virtual int area() = 0;
};
 ...
int main () {
 CPolygon *rect = new CRectangle(4,5);
 CPolygon *trgl = new CTriangle(4,5);
// illegal
//CPolygon *poly = new CPolygon(4,5);
 cout << rect->area() << endl;
 cout << trgl->area() << endl;
 //cout << poly->area() << endl;
 return 0;
}
```

Java

- **In file Polygon.java:**

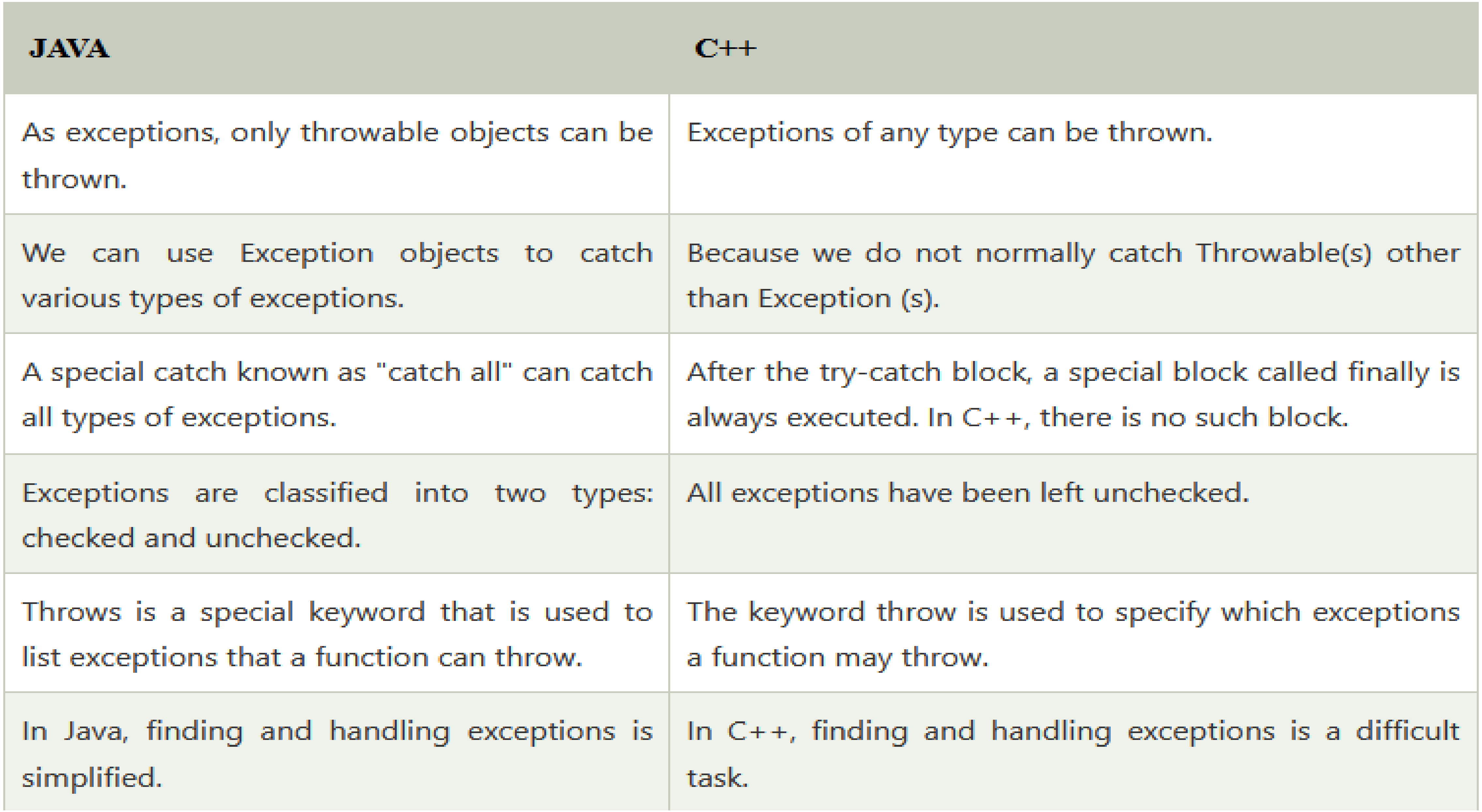
```
public abstract class Polygon {
   protected int width, height;
   public Polygon(int a, int b)
            { width=a; height=b;}
    public abstract int area();
}
```

...

- **In file PolyTest.java:**

```
public class PolyTest {
    public static void int main () {
    Polygon rect = new Rectangle(4,5);
    Polygon trgl = new Triangle(4,5);
    // illegal
    // Polygon poly = new Polygon(4,5);
    System.out.println(rect.area());
    System.out.println(trgl.area());
    // System.out.println(poly.area());
}
}
```

Exceptional Handling



JAVA	C++
As exceptions, only throwable objects can be thrown.	Exceptions of any type can be thrown.
We can use Exception objects to catch various types of exceptions.	Because we do not normally catch Throwable(s) other than Exception (s).
A special catch known as "catch all" can catch all types of exceptions.	After the try-catch block, a special block called finally is always executed. In C++, there is no such block.
Exceptions are classified into two types: checked and unchecked.	All exceptions have been left unchecked.
Throws is a special keyword that is used to list exceptions that a function can throw.	The keyword throw is used to specify which exceptions a function may throw.
In Java, finding and handling exceptions is simplified.	In C++, finding and handling exceptions is a difficult task.