Constructor Types

	Construct from raw material	Construct by assembly

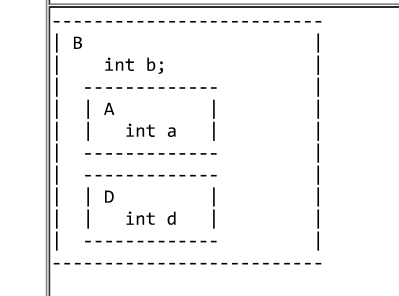
---------- 	
 | Person |
 | int age; |
 ---------- 	
 	
 Complete code
	class Person {
	 print int age=0;
	 public Person(int a1) {
	 age = a1; }
	 }
		class Person {
		 print int age=0;
		 public Person(Person obj) {
		 age = obj.getAge() ; }
		 }

---------- 	
 | Person |
 | int handLength[]; |
 ---------- 	
 	
 Complete code
	class Person {
	 private int handLengths[] = {5, 5};
	 public Person(int handLengths1[]) {
	 handLengths=null;
	 handLengths=(int []).handLengths1.clone();
	 }
	 }
		class Person {
		 private int handLengths[] = {5, 5};
		 public Person(Person obj) {
		 handLengths=null;
		 // It is a bad idea not to clone
		 // handLengths=(int []).obj.geHandLengths();
		 handLengths=(int []).obj.geHandLengths().clone();
		 }
		 }

---------- 	
 | Person |
 | Vector kidAges; |
 ---------- 	
 	
 Complete code
	class Person implements Cloneable
	 {
	 private Vector kidAges;
	
	 public Person(Vector kidAges1) {
	 kidAges=(Vector)kidAges1.clone();
	 }
	 }
		class Person implements Cloneable
		 {
		 private Vector kidAges;
		
		 public Person(Person obj) {
		 // It is a bad idea not to clone
		 // kidAges=(Vector)obj.getKidAges();
		 kidAges=(Vector)obj.getKidAges().clone();
		 }
		 }

---------- 	
 | B |
 | int b; |
 | ---------- |
 | | A | |
 | | int a | |
 | ---------- |
 ---------- 	
 	
 Complete code (toSting() + clone() + equals())
	class A {
	 private int a=0;
	 public A(int a1) { a = a1; }
	 }
	 class B {
	 private int b=0;
	 private A aObj = new A(0);
	 public B(int a1, int b1) {
	 b = b1; aObj.setA(a1); }
	 }
		class A {
		 private int a=0;
		 public A(int a1) { a = a1; }
		 }
		 class B {
		 private int b=0;
		 private A aObj = new A(0);
		 public B(A aObj1, int b1) {
		 b = b1;
		 // Another approach:
		 // If A implements clone(), then
		 // aObj=aObj1.clone();
		 aObj.setA(aObj1.getA());
		 }
		 public B(B bObj1) {
		 b = bObj1.getB();
		 // Another approach:
		 // If A implements clone(), then
		 // aObj=aObj1.clone();
		 aObj.setA(bObj1.getAObj().getA());
		 }
		 public B(A aObj1, B bObj1) {
		 b = bObj1.getB();
		 // Another approach:
		 // If A implements clone(), then
		 // aObj=aObj1.clone();
		 aObj.setA(aObj1.getA());
		 }
		 }

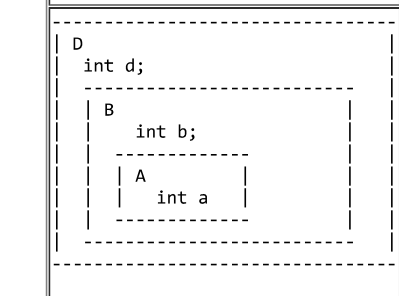
---------- 	
 | B |
 | int b; |
 | ---------- |
 | | A | |
 | | int a,e | |
 | ---------- |
 ---------- 	
 	
 Complete code
	class A {
	 private int a=0, e=0;
	 public A(int a1, int e1) {
	 a = a1; e=e1;}
	 }
	 class B {
	 // Attributes
	 private int b=0;
	 private A aObj = new A(0, 0);
	
	 // Manager functions
	 public B(int a1, int b1, int e1) {
	 b = b1;
	 aObj.setA(a1);
	 aObj.setE(e1);
	 }
	 }
		class A {
		 private int a=0, e=0;
		 public A(int a1, int e1) {
		 a = a1; e=e1;}
		 }
		 class B {
		 // Attributes
		 private int b=0;
		 private A aObj = new A(0, 0);
		
		 // Manager functions
		 public B(B bObj1) {
		 b = bObj1.getB();
		 // Another approach:
		 // If A implements clone(), then
		 // aObj=aObj1.clone();
		 aObj.setA(bObj1.getAObj().getA());
		 aObj.setE(bObj1.getAObj().getE());
		 }
		 public B(A aObj1, B bObj1) {
		 b = bObj1.getB();
		 aObj=null;
		 // Another approach:
		 // If A implements clone(), then
		 // aObj=aObj1.clone();
		 aObj = new A(aObj1.getA(),
		 aObj1.getE());



Complete code (toString() + clone() + equals()

```
class A
{
   private int a=0;
   public A(int a1) { a = a1; }
}
class D
{
   private int d=0;
   public D(int d1) { d = d1; }
}
class B
{
   private int b=0;
   private A aObj = new A(0);
   private D dObj = new D(0);

   public B(int a1, int b1, int d1) {
      b = b1;

      // Approach 1
      // aObj=null;
      // aObj = new A(a1);
      // dObj=null;
      // dObj = new D(d1);

      // Approach 2
      aObj.setA(a1); dObj.setD(d1);
   }
}
```

```
class A
{
   private int a=0;
   public A(int a1) { a = a1; }
}
class D
{
   private int d=0;
   public D(int d1) { d = d1; }
}
class B
{
   private int b=0;
   private A aObj = new A(0);
   private D dObj = new D(0);

   public B(B bObj1) {
      b = bObj1.getB();
      // Another approach:
      // If A implements clone(), then
      //   aObj=aObj1.clone();
      aObj.setA(bObj1.getAObj().getA());
      dObj.setD(bObj1.getDObj().getD());
   }
   public B(A aObj1, B bObj1, D dObj1) {
      b = bObj1.getB();
      // Another approach:
      // If A & D implements clone(), then
      //   aObj=aObj1.clone();
      //   dObj=dObj1.clone();
      aObj.setA(aObj1.getA());
      dObj.setD(dObj1.getD());
   }
}
```



Complete code (toString() + clone() + equals())

```
class A {
   private int a=0;
   public A(int a1) { a = a1; }
}
class B {
   private int b=0;
   private A aObj = new A(0);
   public B(int a1, int b1) {
     b = b1; aObj.setA(a1);
   }
}
class D {
   private int d=0;
   private B bObj = new B(0, 0);

   public D(int a1, int b1, int d1)
   {
     d = d1;

     // Approach 1
     dObj=null;
     dObj = new D(a1, b1, d1);

     // Approach 2
     // bObj.setB(b1);
     // bObj.getAObj().setA(a1);
   }
}
```

```
class A {
   private int a=0;
   public A(int a1) { a = a1; }
}
class B {
   private int b=0;
   private A aObj = new A(0);

   // Constructor 1
   public B(int a1, int b1) {
      b = b1; aObj.setA(a1);
   }
   // Constructor 2
   public B(A aObj1, int b1) {
      b = b1; aObj.setA(aObj1.getA());
   }
   // Constructor 3
   public B(B bObj1) {
      b = bObj1.getB();
      // Another approach:
      // If A implements clone(), then
      //   aObj=bObj1.getA().clone();
      aObj.setA(bObj1.getAObj().getA());
   }
}
class D {
   private int d=0;
   private B bObj = new B(0, 0);


   public D(D dObj1)
   { d = dObj1.getD();
     bObj=null;

     // Another approach:
     // If B implements clone(), then
     //   bObj = bObj1.clone();

     // Use Contructor 1
     bObj = new B(dObj1.getBObj().getAObj().getA(),
                  dObj1.getBObj().getB());

     // Use Contructor 2
     bObj = new B(dObj1.getBObj().getAObj(),
                  dObj1.getBObj().getB());

     // Use Contructor 3
     bObj = new B(dObj1.getBObj());
   }
   public D(B bObj1, int d1)
   { d = d1;
     bObj=null;
```

```
    // Another approach:
    // If B implements clone(), then
    //   bObj = bObj1.clone();

    // Use Contructor 1
    bObj = new B(bObj1.getAObj().getA(),
                 bObj1.getB());

    // Use Contructor 2
    bObj = new B(bObj1.getAObj(),
                 bObj1.getB());

    // Use Contructor 3
    bObj = new B(bObj1);
  }
}
```

```
-------------------------
| Person                |
|                       |
|  -------------        |
|  | Pet       | ..... |
|  |   int age | ..... |
|  -------------        |
-------------------------
```

Complete code

```
class Pet implements Cloneable
{
   // Data members
   private int age=0;

   // Manager function
   public Pet(int age1) {
      age=age1;
   }
   public Object clone() {
      try
      {
         return super.clone();
      }
      catch (CloneNotSupportedException e)
      {
         // This shouldn't happen,
         // since we are Cloneable
         return null;
      }
   }
}
class Person implements Cloneable
{
   private Vector pets;

   // Helping function
   private void trace(String s) {
      System.out.println(s);
   }

   // Manager function
   public Person(Vector pets1) {
      pets=(Vector)pets1.clone();
   }
   public Object clone()
   {
      try
      { Person e = (Person)super.clone();
        e.pets = (Vector)pets.clone();
        /*
        for (int i=0; i < pets.size(); i++) {
          e.pets.setElementAt(
            ((Pet)pets.elementAt(i)).clone(), i);
        }
        */
        return e;
      }
      catch (CloneNotSupportedException e)
      {
        // This shouldn't happen,
        // since we are Cloneable
        return null;
      }
   }
}
```

Notes:

Back | Next

Last modified on: 10/05/2023 20:22:32