```
                                                              --------------2
public interface Instantiable {              ….. 1
      public boolean createObject();           ….. 1
}



public class Attribute {                                  ----------------------4
      private String name;
      private String type;
      private int value;
      private int size;

      Attribute(String n, String t, int v, int s){
            name = n; type = t; value = v; size = s;              ….. 1
      }

      Attribute(Attribute a){             ….. 2
            name = a.name;
            type = a.type;
            value = a.value;
            size = a.size;
      }

      public String getType() {           ….. 0.5
            return type;
      }

      public int getSize() {              ….. 0.5
            return size;
      }


}


                                  --------------27
public abstract class Class implements Instantiable{       ….. 1+1

      private String name;
      private Attribute arrAttr[];              ….. 1
      private int nbAttr;                 ….. 1

      Class(String n, int s){                     ----------------2
            name = n;
            arrAttr = new Attribute[s];                 ….. 1
            nbAttr = 0;                   ….. 1

      }
```

```
	Class(Class c){                                    --------------4
		name = c.name;
		arrAttr = new Attribute[c.arrAttr.length];          ….. 1
		for(int i=0; i<c.nbAttr;i++)                ….. 1
			arrAttr[i] = new Attribute(c.arrAttr[i]);           ….. 1

		nbAttr = c.nbAttr;          ….. 1
	}

	public boolean addAttribute(Attribute a){          -------------3
		if(nbAttr < arrAttr.length){                ….. 1
			arrAttr[nbAttr++] = new Attribute(a);       ….. 1
			return true;                ….. 0.5

		}

		return false;                       ….. 0.5

	}

	public int getObjectSize(){                        --------------3
		int objSize =0;             ….. 0.5
		for(int i=0;i<nbAttr;i++)           ….. 1
			objSize += arrAttr[i].getSize();        ….. 1

		return objSize;             ….. 0.5
	}

	public int countAttr(String type){                 --------------4
		int count=0;            ….. 0.5
		for(int i=0;i<nbAttr;i++){          ….. 1
			if(arrAttr[i].getType().equals(type))       ….. 1
				count++;            ….. 1
		}

		return count;               ….. 0.5

	}

	public Attribute[] getAttributes(String type){     -------------7
		int count = countAttr(type);                ….. 1
		Attribute a[] = new Attribute[count];       ….. 1
		int nbA = 0;            ….. 0.5

		for(int i=0;i<nbAttr;i++){          ….. 1
			if(arrAttr[i].getType().equals(type))       ….. 1
				a[nbA++] = arrAttr[i];              ….. 1+1
		}

		return a;               ….. 0.5
	}

}
```

```
                                                    -------------7
public class AbstractClass extends Class {                  ….. 1
        private int noOfAbsMethods;

        AbstractClass(String n, int s, int nOfAbsM){
                super(n,s);                 ….. 1
                noOfAbsMethods = nOfAbsM;           ….. 1
        }

        AbstractClass(AbstractClass a){
                super(a);                   ….. 1
                noOfAbsMethods = a.noOfAbsMethods;                  ….. 1
        }

        public int getNoOfAbsMethods() {            ….. 1
                return noOfAbsMethods;
        }

        public boolean createObject(){              ….. 1
                return true;
        }
}



public class Package {                      -------45
        public String name;
        public Class arrClasses[];          ….. 1
        public int nbClasses;               ….. 1
        public Package subPackages[];               ….. 1
        public int nbSP;                    ….. 1

        Package(String n, int s){           ---------------4
                name = n;                   ….. 0.5
                arrClasses = new Class[s];                  ….. 1
                nbClasses = 0;              ….. 1
                subPackages = new Package[100];             ….. 1
                nbSP = 0;                   ….. 0.5

        }

        Package(Package p) {                ------------12
                this.name = p.name;                 ….. 0.5
                arrClasses = new Class[p.arrClasses.length];        ….. 1
                nbClasses = p.nbClasses;            ….. 0.5

                for(int i=0;i<nbClasses;i++) {              ….. 1
                if (p.arrClasses[i] instanceof AbstractClass)           ….. 1
                        arrClasses[i] = new AbstractClass(              ….. 1
                                        (AbstractClass)p.arrClasses[i]);        ….. 1
                else
                        arrClasses[i] = new ConcreteClass(              ….. 1
                                (ConcreteClass)p.arrClasses[i]);                    ….. 1
```

```
        }

        subPackages = new Package[p.subPackages.length];           …. 1
        nbSP = p.nbSP;              …. 1
        for(int j=0; j<nbSP; j++) {                    …. 1
              subPackages[j] = new Package(p.subPackages[j]);  …. 1
        }
}

public boolean addClass(Class c){              ------------9
        if(nbClasses < arrClasses.length){             …. 1
              if(c instanceof AbstractClass)           …. 1
                    arrClasses[nbClasses++] = new          …. 1
                          AbstractClass((AbstractClass)c);       …. 2
              else
                    arrClasses[nbClasses++] = new          …. 1
                          ConcreteClass((ConcreteClass)c);       …. 2
        return true;          …. 0.5
        }
        return false;         …. 0.5
}

public int getPackageSize(){                  --------------5
        int totalPackageSize = 0;              …. 0.5

        for(int i=0; i<nbClasses;i++) {        …. 1
              totalPackageSize += arrClasses[i].getObjectSize();         …. 1
        }

        for(int j=0;j<nbSP;j++) {             …. 1
              totalPackageSize += subPackages[j].getPackageSize();       …. 1
        }
        return totalPackageSize;          …. 0.5

}

public AbstractClass[] getAndFill(ConcreteClass[] conClass, int n, int s){--11
        AbstractClass a[] = new AbstractClass[arrClasses.length];        …. 1
        int nbA = 0;          …. 0.5
        int nbCC = 0;               …. 0.5
        for(int i=0;i<nbClasses;i++){          …. 1
              if(arrClasses[i] instanceof ConcreteClass){         …. 1
                if(arrClasses[i].getObjectSize() > s)             …. 1
                          conClass[nbCC++] = (ConcreteClass)arrClasses[i]; ….1
              }
              else {
                if(((AbstractClass)arrClasses[i]).getNoOfAbsMethods()==n) …. 2
                          a[nbA++] = (AbstractClass)arrClasses[i]; …. 2
              }
        }

        return a;             …. 1
}}
```