第一题：

*/\*\*  
 \** ***@author*** *WangMingMing  
 \** ***@creat*** *2020-03-16 14:29  
 \* 任务接口  
 \*/*public interface Task {  
 */\*\*  
 \* 执行具体任务的接口方法  
 \*/* public abstract void execute();  
  
}

*/\*\*  
 \** ***@author*** *WangMingMing  
 \** ***@creat*** *2020-03-16 14:30  
 \* 任务服务接口  
 \*/*public interface TaskService {  
 */\*\*  
 \* 执行任务接口列表中的每个任务  
 \*/* public void executeTasks();  
  
 */\*\*  
 \* 添加任务  
 \** ***@param*** *t 新添加的任务  
 \*/* public void addTask(Task t);  
  
}

import java.util.ArrayList;  
import java.util.List;  
  
*/\*\*  
 \** ***@author*** *WangMingMing  
 \** ***@creat*** *2020-03-16 14:33  
 \*/*public class TaskServiceImpl implements TaskService{  
 List<Task> listTasks = new ArrayList<Task>();  
  
 public static void main(String[] args) {  
 //首先测试三个具体类  
 System.*out*.println("-------下列输出: 测试实现Task接口的三个具体类-------");  
 FirstClass firstClass = new FirstClass();  
 firstClass.execute();  
 SecondClass secondClass = new SecondClass();  
 secondClass.execute();  
 ThirdClass thirdClass = new ThirdClass();  
 thirdClass.execute();  
 System.*out*.println("-------------------------------------------------");  
 System.*out*.println();  
 //测试实现TaskService接口的类  
 System.*out*.println("-------下列输出: 测试实现TaskService接口的具体类------");  
 TaskServiceImpl taskServiceImpl = new TaskServiceImpl();  
 taskServiceImpl.execute();  
 //将上述的三个具体类添加  
 taskServiceImpl.addTask(firstClass);  
 taskServiceImpl.addTask(secondClass);  
 taskServiceImpl.addTask(thirdClass);  
 System.*out*.println("----下面是将三个具体类对象添加到List之后打印List里面的内容----");  
 taskServiceImpl.executeTasks();  
  
  
 }  
 public void execute(){  
 System.*out*.println("这是第四个具体类TaskServiceImpl的任务执行");  
 }  
 public void executeTasks(){  
 for(Task t : listTasks){  
 System.*out*.println(t.toString());  
 }  
 }  
 public void addTask(Task t){  
 listTasks.add(t);  
 }  
}  
  
  
  
class FirstClass implements Task{  
 public FirstClass(){  
  
 }  
 public void execute(){  
 System.*out*.println("这是第一个具体类FirstClass的任务执行");  
 }  
  
 @Override  
 public String toString() {  
 return "打印第一个类";  
 }  
}  
  
class SecondClass implements Task{  
 public SecondClass(){  
  
 }  
 public void execute(){  
 System.*out*.println("这是第二个具体类SecondClass的任务执行");  
 }  
  
 @Override  
 public String toString() {  
 return "打印第二个类";  
 }  
}  
class ThirdClass implements Task{  
 public ThirdClass(){  
  
 }  
 public void execute(){  
 System.*out*.println("这是第三个具体类ThirdClass的任务执行");  
 }  
  
 @Override  
 public String toString() {  
 return "打印第三个类";  
 }  
}

第二题：

import java.sql.Struct;  
import java.util.ArrayList;  
import java.util.List;  
  
*/\*\*  
 \** ***@author*** *WangMingMing  
 \** ***@creat*** *2020-03-16 16:01  
 \*/*public class Course implements Cloneable{  
 private String courseName;  
 private List<Person> students = new ArrayList<>();  
 private Person teacher;  
  
 public Course(String courseName, Person teacher) {  
 this.courseName = courseName;  
 this.teacher = teacher;  
 }  
  
 public void register(Person s){  
 if(!students.isEmpty() && students.contains(s)){  
 System.*out*.println("该学生对象已经存在");  
 }else{  
 students.add(s);  
 }  
 }  
  
 public String getCourseName() {  
 return courseName;  
 }  
  
 public List<Person> getStudents() {  
 return students;  
 }  
  
 public Person getTeacher() {  
 return teacher;  
 }  
 public void unregister(Person s){  
 //如果不存在  
 if(!students.contains(s)){  
 //暂时什么也不做  
 }else{  
 students.remove(s);  
 }  
 }  
 public int getNumberOfStudent(){  
 return students.size();  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 /\*if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 Course course = (Course) obj;  
 return courseName.equals(course.courseName) &&  
 (course.students.size() == students.size() && students.containsAll(course.students)) &&  
 teacher.equals(course.teacher);\*/  
 if(obj instanceof Course){  
 Course course = (Course)obj;  
 return courseName.equals(course.courseName) &&  
 (course.students.size() == students.size() && students.containsAll(course.students)) &&  
 teacher.equals(course.teacher);  
 }  
 return false;  
 }  
  
  
  
 @Override  
 public Object clone() throws CloneNotSupportedException {  
 Course newObj = (Course)(super.clone());  
 newObj.teacher = (Faculty)(this.teacher.clone());  
 newObj.courseName = new String(this.getCourseName());  
 List<Person> list = new ArrayList<Person>();  
 for(Person s : this.students){  
 Student student = new Student();  
 student = (Student)s.clone();  
 list.add(student);  
 }  
 newObj.students = list;  
 return newObj;  
 }  
  
  
 @Override  
 public String toString() {  
 String str = new String();  
 str = "Course Name: " + this.getCourseName() + "\n" +  
 this.teacher.toString() + "Student List: \n";  
 for(Person s : this.students){  
 str += s.toString();  
 }  
 str += "Totally: " + this.getNumberOfStudent() + " students.\n";  
 return str;  
 }  
}

*/\*\*  
 \** ***@author*** *WangMingMing  
 \** ***@creat*** *2020-03-16 16:50  
 \*/*public class CourseTest {  
 public static void main(String[] args) throws CloneNotSupportedException{  
 Person teacher = new Faculty("James Gosling", 65, 0000, "Professor", "http://...");  
 Course javaCourse = new Course("Java Language Programming", teacher);  
 Person student1 = new Student("aaa", 20, 20170101, "CS", "CS1704");  
 Person student2 = new Student("bbb", 20, 20170102, "CS", "CS1705");  
 Person student3 = new Student("ccc", 20, 20170103, "CS", "CS1706");  
 javaCourse.register(student1);  
 javaCourse.register(student2);  
 javaCourse.register(student3);  
 System.*out*.println(javaCourse);  
 javaCourse.unregister(student3);  
 System.*out*.println(javaCourse);  
 //测试是否为深拷贝  
 Course javaCourse2 = (Course)javaCourse.clone();  
 //javaCourse2.getTeacher().setAge(30);  
 //javaCourse2.getTeacher().setName("ggggggg");  
 //((Faculty)(javaCourse2.getTeacher())).setFacultyId(100);  
 System.*out*.println(javaCourse.equals(javaCourse2));  
 System.*out*.println(javaCourse.getCourseName() != javaCourse2.getCourseName());  
 System.*out*.println(javaCourse.getTeacher() != javaCourse2.getTeacher());  
 System.*out*.println(javaCourse.getStudents() != javaCourse2.getStudents());  
 System.*out*.println(javaCourse2);  
 }  
}

import java.util.Objects;  
  
*/\*\*  
 \** ***@author*** *WangMingMing  
 \** ***@creat*** *2020-03-16 16:08  
 \*/*public class Faculty extends Person implements Cloneable{  
 private int facultyId;  
 private String title;  
 private String email;  
  
 public Faculty() {  
  
 }  
  
 public Faculty(String name, int age, int facultyId, String title, String email) {  
 super(name, age);  
 this.facultyId = facultyId;  
 this.title = title;  
 this.email = email;  
 }  
  
 public int getFacultyId() {  
 return facultyId;  
 }  
  
 public void setFacultyId(int facultyId) {  
 this.facultyId = facultyId;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 /\*if(super.equals(obj)){  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 if (!super.equals(obj)) return false;  
 Faculty faculty = (Faculty) obj;  
 return facultyId == faculty.facultyId &&  
 Objects.equals(title, faculty.title) &&  
 Objects.equals(email, faculty.email);  
 }\*/  
 if(super.equals(obj)){  
 if(obj instanceof Faculty){  
 Faculty faculty = (Faculty) obj;  
 return facultyId == faculty.facultyId &&  
 Objects.*equals*(title, faculty.title) &&  
 Objects.*equals*(email, faculty.email);  
 }  
 return false;  
 }  
 return false;  
 }  
  
 @Override  
 public Object clone() throws CloneNotSupportedException {  
 Faculty newObj = (Faculty)(super.clone());  
 newObj.facultyId = this.getFacultyId();  
 newObj.title = new String(this.getTitle());  
 newObj.email = new String(this.getEmail());  
 return newObj;  
 }  
  
 @Override  
 public String toString() {  
 String str = "Teacher Info: ";  
 str += super.toString();  
 return str + ", facultyId: " + this.getFacultyId() +  
 ", title: " + this.getTitle() +  
 ", email: " + this.getEmail() + "\n";  
 }  
}

import java.util.Objects;  
  
*/\*\*  
 \** ***@author*** *WangMingMing  
 \** ***@creat*** *2020-03-16 15:58  
 \*/*public class Person implements Cloneable{  
 private int age;  
 private String name;  
  
 public Person(){  
  
 }  
 public Person(String name, int age) {  
 this.age = age;  
 this.name = name;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 @Override  
 public String toString() {  
 return "name: " + this.getName() + ", age: " + this.getAge();  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 /\*if (obj == null || this.getClass() != obj.getClass()) return false;  
 if (this == obj) return true;  
 Person person = (Person) obj;\*/  
 if(obj instanceof Person){  
 Person person = (Person) obj;  
 return this.getAge() == person.getAge() && Objects.*equals*(this.getName(), person.getName());  
 }  
 return false;  
  
 }  
  
  
 @Override  
 public Object clone() throws CloneNotSupportedException {  
 Person newObj = (Person)(super.clone());  
 newObj.age = this.getAge();  
 newObj.name = new String(this.getName());  
 return newObj;  
 }  
}

import java.util.Objects;  
  
*/\*\*  
 \** ***@author*** *WangMingMing  
 \** ***@creat*** *2020-03-16 16:11  
 \*/*public class Student extends Person implements Cloneable{  
 private int studentId;  
 private String department;  
 private String classNo;  
 public Student(){  
  
 }  
  
 public Student(String name, int age, int studentId, String department, String classNo) {  
 super(name, age);  
 this.studentId = studentId;  
 this.department = department;  
 this.classNo = classNo;  
 }  
  
 public int getStudentId() {  
 return studentId;  
 }  
  
 public void setStudentId(int studentId) {  
 this.studentId = studentId;  
 }  
  
 public String getDepartment() {  
 return department;  
 }  
  
 public void setDepartment(String department) {  
 this.department = department;  
 }  
  
 public String getClassNo() {  
 return classNo;  
 }  
  
 public void setClassNo(String classNo) {  
 this.classNo = classNo;  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if(super.equals(obj)){  
 if(obj instanceof Student){  
 Student student = (Student)obj;  
 return studentId == student.studentId &&  
 Objects.*equals*(department, student.department) &&  
 Objects.*equals*(classNo, student.classNo);  
 }  
 return false;  
 }  
 return false;  
 }  
  
 @Override  
 public Object clone() throws CloneNotSupportedException {  
 Student newObj = (Student)(super.clone());  
 newObj.studentId = this.getStudentId();  
 newObj.classNo = new String(this.getClassNo());  
 newObj.department = new String(this.getDepartment());  
 return newObj;  
  
 }  
  
 @Override  
 public String toString() {  
 String str = "";  
 str += super.toString();  
 return str + ", studentId: " + this.getStudentId() +  
 ", department: " + this.getDepartment() +  
 ", classNo: " + this.getClassNo() + "\n";  
 }  
}

第三题：

*/\*\*  
 \** ***@author*** *WangMingMing  
 \** ***@creat*** *2020-03-17 8:46  
 \*/*public class AtomicComponent extends Component{  
 public AtomicComponent() {  
 }  
  
 public AtomicComponent(int id, String name, double price) {  
 super(id, name, price);  
 }  
 public void add(Component component) throws UnsupportedOperationException{  
 throw new UnsupportedOperationException();  
 }  
 public double calcPrice(){  
 this.setPrice(this.price);  
 return this.price;  
 }  
 public Iterator iterator(){  
 return new NullIterator();  
 }  
 public void remove(Component component) throws UnsupportedOperationException{  
 throw new UnsupportedOperationException();  
 }  
  
}

import java.util.Objects;  
  
*/\*\*  
 \** ***@author*** *WangMingMing  
 \** ***@creat*** *2020-03-17 8:38  
 \*/*public abstract class Component {  
 protected int id;  
 protected String name;  
 protected double price;  
 public Component(){  
  
 }  
  
 public Component(int id, String name, double price) {  
 this.id = id;  
 this.name = name;  
 this.price = price;  
 }  
 public abstract void add(Component component);  
 public abstract double calcPrice();  
 public abstract void remove(Component component);  
 public abstract Iterator iterator();  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public double getPrice() {  
 this.price = calcPrice();  
 return this.price;  
 }  
  
 public void setPrice(double price) {  
 this.price = price;  
 }  
  
 @Override  
 public String toString() {  
 return "id: " + this.getId() +  
 ", name: " + this.getName() +  
 ", price: " + this.getPrice() + "\n";  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 Component component = (Component) obj;  
 return id == component.id &&  
 Double.*compare*(component.price, price) == 0 &&  
 Objects.*equals*(name, component.name);  
 }  
  
  
}

*/\*\*  
 \** ***@author*** *WangMingMing  
 \** ***@creat*** *2020-03-17 9:10  
 \*/*public class ComponentFactory {  
 public ComponentFactory(){  
  
 }  
 public static Component creat(){  
 int id = 0;  
 Component computer = new CompositeComponent(id++, "Think pad", 0.0);  
 Component keyboard = new AtomicComponent(id++, "Keyboard", 20.0);  
 Component mouse = new AtomicComponent(id++, "Mouse", 20.0);  
 Component monitor = new AtomicComponent(id++, "Monitor", 1000.0);  
 computer.add(keyboard);  
 computer.add(mouse);  
 computer.add(monitor);  
  
 Component mainFrame = new CompositeComponent(id++, "Main frame", 0.0);  
 Component hardDisk = new AtomicComponent(id++, "Hard disk", 1000);  
 Component powerSupplier = new AtomicComponent(id++, "Power supplier", 500);  
 mainFrame.add(hardDisk);  
 mainFrame.add(powerSupplier);  
  
 Component mainBoard = new CompositeComponent(id++, "Main board", 0.0);  
 Component cpu = new AtomicComponent(id++, "CPU", 1500.0);  
 Component videoCard = new AtomicComponent(id++, "Video card", 900);  
 Component networkCard = new AtomicComponent(id++, "Network card", 100);  
 mainBoard.add(cpu);  
 mainBoard.add(videoCard);  
 mainBoard.add(networkCard);  
  
 mainFrame.add(mainBoard);  
 computer.add(mainFrame);  
 return computer;  
 }  
}

import java.util.ArrayList;  
  
*/\*\*  
 \** ***@author*** *WangMingMing  
 \** ***@creat*** *2020-03-17 9:08  
 \*/*public class ComponentList extends ArrayList<Component> implements Iterator {  
 private int position = -1;  
 public ComponentList(){  
  
 }  
 public boolean hasNext(){  
 return position + 1 < this.size();  
 }  
 public Component next(){  
 if(this.hasNext()){  
 position++;  
 return this.get(position);  
 }else{  
 return null;  
 }  
 }  
  
  
  
}

*/\*\*  
 \** ***@author*** *WangMingMing  
 \** ***@creat*** *2020-03-17 9:03  
 \*/*public class CompositeComponent extends Component {  
 protected ComponentList childs = new ComponentList();  
  
 public CompositeComponent() {  
 }  
  
 public CompositeComponent(int id, String name, double price) {  
 super(id, name, price);  
 }  
 public void add(Component component) throws UnsupportedOperationException{  
 childs.add(component);  
 }  
 public double calcPrice(){  
 double price = 0;  
 for(Component c : this.childs){  
 price += c.getPrice();  
 }  
 return price;  
 }  
  
 public Iterator iterator(){  
 return new CompositeIterator(childs);  
 }  
 public void remove(Component component) throws UnsupportedOperationException{  
 childs.remove(component);  
 this.price -= component.getPrice();  
 }  
  
 @Override  
 public double getPrice(){  
 return this.calcPrice();  
 }  
 @Override  
 public String toString() {  
 String str = super.toString();  
 String str1 = "";  
 for(Component c : childs){  
 str1 += c.toString();  
 }  
 return str + "sub-components of " + this.getName() + ": \n" + str1;  
 }  
}

import java.util.ArrayList;  
import java.util.List;  
  
*/\*\*  
 \** ***@author*** *WangMingMing  
 \** ***@creat*** *2020-03-17 9:00  
 \*/*public class CompositeIterator implements Iterator{  
 protected List<Iterator> iterators = new ArrayList<>();  
  
 public CompositeIterator(Iterator iterator) {  
 iterators.add(iterator);  
 }  
 @Override  
 public boolean hasNext(){  
 if(iterators.isEmpty()){  
 return false;  
 }  
 Iterator iterator = iterators.get(iterators.size() - 1);  
 if(!iterator.hasNext()){  
 iterators.remove(iterators.size() - 1);  
 return hasNext();  
 }else{  
 return true;  
 }  
 }  
 @Override  
 public Component next(){  
 if(hasNext()){  
 Iterator iterator = iterators.get(iterators.size() - 1);  
 Component component = (Component)iterator.next();  
 iterators.add(component.iterator());  
 return component;  
 }  
 return null;  
 }  
}

*/\*\*  
 \** ***@author*** *WangMingMing  
 \** ***@creat*** *2020-03-17 8:36  
 \*/*public interface Iterator {  
 public boolean hasNext();  
 public Component next();  
}

*/\*\*  
 \** ***@author*** *WangMingMing  
 \** ***@creat*** *2020-03-17 8:57  
 \*/*public class NullIterator implements Iterator{  
 public NullIterator() {  
 }  
 @Override  
 public boolean hasNext(){  
 return false;  
 }  
 @Override  
 public Component next(){  
 return null;  
 }  
}

*/\*\*  
 \** ***@author*** *WangMingMing  
 \** ***@creat*** *2020-03-17 9:20  
 \*/*public class Test {  
 public static void main(String[] args) {  
 Component computer = ComponentFactory.*creat*();  
 System.*out*.println(computer);  
 System.*out*.println("id: " + computer.getId() + ", name: "+ computer.getName() +  
 ", price: " + computer.getPrice());  
 Iterator it = computer.iterator();  
 while(it.hasNext()){  
 Component c = it.next();  
 System.*out*.println("id: " + c.getId() + ", name: " + c.getName() + ", price: " + c.getPrice());  
 }  
  
 }  
}