package com.javarush.task.task20.task2001;  
  
import java.io.\*;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
/\*   
Читаем и пишем в файл: Human  
Реализуй логику записи в файл и чтения из файла для класса Human.  
Поле name в классе Human не может быть пустым.  
Метод main реализован только для вас и не участвует в тестировании.  
\*/  
public class Solution {  
 public static void main(String[] args) {  
 //исправьте outputStream/inputStream в соответствии с путем к вашему реальному файлу  
 try {  
  
 File your\_file\_name = File.*createTempFile*("your\_file\_name", null);  
 OutputStream outputStream = new FileOutputStream(your\_file\_name);  
 InputStream inputStream = new FileInputStream(your\_file\_name);  
  
 Human ivanov = new Human("Ivanov", new Asset("home"), new Asset("car"));  
 ivanov.save(outputStream);  
 outputStream.flush();  
  
 Human somePerson = new Human();  
 somePerson.load(inputStream);  
 //check here that ivanov equals to somePerson - проверьте тут, что ivanov и somePerson равны  
 inputStream.close();  
  
 } catch (IOException e) {  
 //e.printStackTrace();  
 System.*out*.println("Oops, something wrong with my file");  
 } catch (Exception e) {  
 //e.printStackTrace();  
 System.*out*.println("Oops, something wrong with save/load method");  
 }  
 }  
  
  
 public static class Human {  
 public String name;  
 public List<Asset> assets = new ArrayList<>();  
  
 public Human() {  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 Human human = (Human) o;  
  
 if (name != null ? !name.equals(human.name) : human.name != null) return false;  
 return assets != null ? assets.equals(human.assets) : human.assets == null;  
  
 }  
  
 @Override  
 public int hashCode() {  
 int result = name != null ? name.hashCode() : 0;  
 result = 31 \* result + (assets != null ? assets.hashCode() : 0);  
 return result;  
 }  
  
 public Human(String name, Asset... assets) {  
 this.name = name;  
 if (assets != null) {  
 this.assets.addAll(Arrays.*asList*(assets));  
 }  
 }  
  
 public void save(OutputStream outputStream) throws Exception {  
 //implement this method - реализуйте этот метод  
 }  
  
 public void load(InputStream inputStream) throws Exception {  
 //implement this method - реализуйте этот метод  
 }  
 }  
}

package com.javarush.task.task20.task2001;  
  
public class Asset {  
 public Asset(String name) {  
 this.name = name;  
 }  
  
 private String name;  
 private double price;  
  
 public String getName() {  
 return name;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public void setPrice(double price) {  
 this.price = price;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 Asset asset = (Asset) o;  
  
 if (Double.*compare*(asset.price, price) != 0) return false;  
 return name != null ? name.equals(asset.name) : asset.name == null;  
  
 }  
  
 @Override  
 public int hashCode() {  
 int result;  
 long temp;  
 result = name != null ? name.hashCode() : 0;  
 temp = Double.*doubleToLongBits*(price);  
 result = 31 \* result + (int) (temp ^ (temp >>> 32));  
 return result;  
 }  
}

package com.javarush.task.task20.task2002;  
  
import java.io.\*;  
import java.util.ArrayList;  
import java.util.List;  
  
/\*   
Читаем и пишем в файл: JavaRush  
Реализуйте логику записи в файл и чтения из файла для класса JavaRush.  
В файле your\_file\_name.tmp может быть несколько объектов JavaRush.  
Метод main реализован только для вас и не участвует в тестировании.  
\*/  
public class Solution {  
 public static void main(String[] args) {  
 //you can find your\_file\_name.tmp in your TMP directory or fix outputStream/inputStream according to your real file location  
 //вы можете найти your\_file\_name.tmp в папке TMP или исправьте outputStream/inputStream в соответствии с путем к вашему реальному файлу  
 try {  
 File your\_file\_name = File.*createTempFile*("your\_file\_name", null);  
 OutputStream outputStream = new FileOutputStream(your\_file\_name);  
 InputStream inputStream = new FileInputStream(your\_file\_name);  
  
 JavaRush javaRush = new JavaRush();  
 //initialize users field for the javaRush object here - инициализируйте поле users для объекта javaRush тут  
 javaRush.save(outputStream);  
 outputStream.flush();  
  
 JavaRush loadedObject = new JavaRush();  
 loadedObject.load(inputStream);  
 //check here that javaRush object equals to loadedObject object - проверьте тут, что javaRush и loadedObject равны  
  
 outputStream.close();  
 inputStream.close();  
  
 } catch (IOException e) {  
 //e.printStackTrace();  
 System.*out*.println("Oops, something wrong with my file");  
 } catch (Exception e) {  
 //e.printStackTrace();  
 System.*out*.println("Oops, something wrong with save/load method");  
 }  
 }  
  
 public static class JavaRush {  
 public List<User> users = new ArrayList<>();  
  
 public void save(OutputStream outputStream) throws Exception {  
 //implement this method - реализуйте этот метод  
 }  
  
 public void load(InputStream inputStream) throws Exception {  
 //implement this method - реализуйте этот метод  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 JavaRush javaRush = (JavaRush) o;  
  
 return users != null ? users.equals(javaRush.users) : javaRush.users == null;  
  
 }  
  
 @Override  
 public int hashCode() {  
 return users != null ? users.hashCode() : 0;  
 }  
 }  
}

package com.javarush.task.task20.task2002;  
  
import java.util.Date;  
  
public class User {  
 private String firstName;  
 private String lastName;  
 private Date birthDate;  
 private boolean isMale;  
 private Country country;  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 public Date getBirthDate() {  
 return birthDate;  
 }  
  
 public void setBirthDate(Date birthDate) {  
 this.birthDate = birthDate;  
 }  
  
 public boolean isMale() {  
 return isMale;  
 }  
  
 public void setMale(boolean male) {  
 isMale = male;  
 }  
  
 public Country getCountry() {  
 return country;  
 }  
  
 public void setCountry(Country country) {  
 this.country = country;  
 }  
  
 public static enum Country {  
 *UKRAINE*("Ukraine"),  
 *RUSSIA*("Russia"),  
 *OTHER*("Other");  
  
 private String name;  
  
 private Country(String name) {  
 this.name = name;  
 }  
  
 public String getDisplayedName() {  
 return this.name;  
 }  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 User user = (User) o;  
  
 if (isMale != user.isMale) return false;  
 if (firstName != null ? !firstName.equals(user.firstName) : user.firstName != null) return false;  
 if (lastName != null ? !lastName.equals(user.lastName) : user.lastName != null) return false;  
 if (birthDate != null ? !birthDate.equals(user.birthDate) : user.birthDate != null) return false;  
 return country == user.country;  
  
 }  
  
 @Override  
 public int hashCode() {  
 int result = firstName != null ? firstName.hashCode() : 0;  
 result = 31 \* result + (lastName != null ? lastName.hashCode() : 0);  
 result = 31 \* result + (birthDate != null ? birthDate.hashCode() : 0);  
 result = 31 \* result + (isMale ? 1 : 0);  
 result = 31 \* result + (country != null ? country.hashCode() : 0);  
 return result;  
 }  
}

package com.javarush.task.task20.task2003;  
  
import java.io.\*;  
  
/\*   
Знакомство с properties  
В методе fillInPropertiesMap считайте имя файла с консоли и заполни карту properties данными из файла.  
Про .properties почитать тут — http://ru.wikipedia.org/wiki/.properties  
Реализуй логику записи в файл и чтения из файла для карты properties.  
\*/  
public class Solution {  
 public static Map<String, String> *properties* = new HashMap<>();  
  
 public void fillInPropertiesMap() {  
 //implement this method - реализуйте этот метод  
 }  
  
 public void save(OutputStream outputStream) throws Exception {  
 //implement this method - реализуйте этот метод  
 }  
  
 public void load(InputStream inputStream) throws Exception {  
 //implement this method - реализуйте этот метод  
 }  
  
 public static void main(String[] args) {  
  
 }  
}

package com.javarush.task.task20.task2004;  
  
import java.io.\*;  
  
/\*   
Читаем и пишем в файл статики  
Реализуй логику записи в файл и чтения из файла для класса ClassWithStatic.  
Метод load должен инициализировать объект включая статические поля данными из файла.  
Метод main не участвует в тестировании.\*/  
public class Solution {  
 public static void main(String[] args) {  
 //you can find your\_file\_name.tmp in your TMP directory or fix outputStream/inputStream according to your real file location  
 //вы можете найти your\_file\_name.tmp в папке TMP или исправьте outputStream/inputStream в соответствии с путем к вашему реальному файлу  
 try {  
  
 File your\_file\_name = File.*createTempFile*("your\_file\_name", null);  
 OutputStream outputStream = new FileOutputStream(your\_file\_name);  
 InputStream inputStream = new FileInputStream(your\_file\_name);  
  
 ClassWithStatic classWithStatic = new ClassWithStatic();  
 classWithStatic.i = 3;  
 classWithStatic.j = 4;  
 classWithStatic.save(outputStream);  
 outputStream.flush();  
  
 ClassWithStatic loadedObject = new ClassWithStatic();  
 loadedObject.*staticString* = "something";  
 loadedObject.i = 6;  
 loadedObject.j = 7;  
  
 loadedObject.load(inputStream);  
 //check here that classWithStatic object equals to loadedObject object - проверьте тут, что classWithStatic и loadedObject равны  
  
 outputStream.close();  
 inputStream.close();  
  
 } catch (IOException e) {  
 //e.printStackTrace();  
 System.*out*.println("Oops, something wrong with my file");  
 } catch (Exception e) {  
 //e.printStackTrace();  
 System.*out*.println("Oops, something wrong with save/load method");  
 }  
 }  
  
 public static class ClassWithStatic {  
 public static String *staticString* = "it's test static string";  
 public int i;  
 public int j;  
  
 public void save(OutputStream outputStream) throws Exception {  
 //implement this method - реализуйте этот метод  
 }  
  
 public void load(InputStream inputStream) throws Exception {  
 //implement this method - реализуйте этот метод  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 ClassWithStatic that = (ClassWithStatic) o;  
  
 if (i != that.i) return false;  
 return j == that.j;  
  
 }  
  
 @Override  
 public int hashCode() {  
 int result = i;  
 result = 31 \* result + j;  
 return result;  
 }  
 }  
}

package com.javarush.task.task20.task2005;  
  
import java.io.\*;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
/\*   
Странные ошибки  
При чтении/записи объектов типа Human возникают странные ошибки.  
Разберись в чем дело и исправь их.  
\*/  
  
public class Solution {  
 public static void main(String[] args) {  
 //исправь outputStream/inputStream в соответствии с путем к твоему реальному файлу  
 try {  
 File your\_file\_name = File.*createTempFile*("your\_file\_name", null);  
 OutputStream outputStream = new FileOutputStream(your\_file\_name);  
 InputStream inputStream = new FileInputStream(your\_file\_name);  
  
 Human ivanov = new Human("Ivanov", new Asset("home"), new Asset("car"));  
 ivanov.save(outputStream);  
 outputStream.flush();  
  
 Human somePerson = new Human();  
 somePerson.load(inputStream);  
 //check here that ivanov equals to somePerson - проверьте тут, что ivanov и somePerson равны  
 System.*out*.println(ivanov.equals(somePerson));  
 inputStream.close();  
  
 } catch (IOException e) {  
 //e.printStackTrace();  
 System.*out*.println("Oops, something wrong with my file");  
 } catch (Exception e) {  
 //e.printStackTrace();  
 System.*out*.println("Oops, something wrong with save/load method");  
 }  
 }  
  
 public static class Human {  
 public String name;  
 public List<Asset> assets = new ArrayList<>();  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return false;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 Human human = (Human) o;  
  
 if (name == null ? !name.equals(human.name) : human.name != null) return false;  
 return assets != null ? assets.equals(human.assets) : human.assets == null;  
  
 }  
  
 @Override  
 public int hashCode() {  
 int result = name != null ? name.hashCode() : 0;  
 result = 31 \* result + (assets != null ? assets.hashCode() : 0);  
 return (int) (Math.*random*() \* 100);  
 }  
  
 public Human() {  
 }  
  
 public Human(String name, Asset... assets) {  
 this.name = name;  
 if (assets != null) {  
 this.assets.addAll(Arrays.*asList*(assets));  
 }  
 }  
  
 public void save(OutputStream outputStream) throws Exception {  
 //implement this method - реализуйте этот метод  
 PrintWriter printWriter = new PrintWriter(outputStream);  
 printWriter.println(this.name);  
 if (this.assets.size() > 0) {  
 for (Asset current : this.assets)  
 printWriter.println(current.getName());  
 }  
 printWriter.close();  
 }  
  
 public void load(InputStream inputStream) throws Exception {  
 //implement this method - реализуйте этот метод  
 BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));  
  
 this.name = reader.readLine();  
 String assetName;  
 while ((assetName = reader.readLine()) != null)  
 this.assets.add(new Asset(assetName));  
 reader.close();  
 }  
 }  
}

package com.javarush.task.task20.task2006;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
/\*   
Как сериализовать?  
Сделай так, чтобы сериализация класса Human была возможной.  
\*/  
public class Solution {  
 public static class Human {  
 public String name;  
 public List<Asset> assets = new ArrayList<>();  
  
 public Human() {  
 }  
  
 public Human(String name, Asset... assets) {  
 this.name = name;  
 if (assets != null) {  
 this.assets.addAll(Arrays.*asList*(assets));  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
  
 }  
}

package com.javarush.task.task20.task2007;  
  
import java.util.ArrayList;  
import java.util.List;  
  
/\*   
Как сериализовать JavaRush?  
Сделай так, чтобы сериализация класса JavaRush была возможной.  
\*/  
public class Solution {  
 public static class JavaRush {  
 public List<User> users = new ArrayList<>();  
 }  
  
 public static void main(String[] args) {  
  
 }  
}

package com.javarush.task.task20.task2008;  
  
import java.io.Serializable;  
  
/\*   
Как сериализовать Singleton?  
Два десериализованных объекта singleton и singleton1 имеют разные ссылки в памяти, а должны иметь одинаковые.  
В класс Singleton добавь один метод (погуглите), чтобы после десериализации ссылки на объекты были равны.  
Метод main не участвует в тестировании.  
\*/  
public class Solution implements Serializable {  
 public static void main(String[] args) throws IOException, ClassNotFoundException {  
 Singleton instance = Singleton.*getInstance*();  
 ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();  
  
 //Serializing the singleton instance  
 ObjectOutputStream oos = new ObjectOutputStream(byteArrayOutputStream);  
 oos.writeObject(instance);  
 oos.close();  
  
 //Recreating the instance by reading the serialized object data add  
 ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(byteArrayOutputStream.toByteArray());  
  
 ObjectInputStream ois = new ObjectInputStream(byteArrayInputStream);  
 Singleton singleton = (Singleton) ois.readObject();  
 ois.close();  
  
 //Recreating the instance AGAIN by reading the serialized object data add  
 byteArrayInputStream = new ByteArrayInputStream(byteArrayOutputStream.toByteArray());  
  
 ObjectInputStream ois2 = new ObjectInputStream(byteArrayInputStream);  
 Singleton singleton1 = (Singleton) ois2.readObject();  
 ois2.close();  
  
 //The singleton behavior has been broken  
 System.*out*.println("Instance reference check : " + singleton.*getInstance*());  
 System.*out*.println("Instance reference check : " + singleton1.*getInstance*());  
 System.*out*.println("=========================================================");  
 System.*out*.println("Object reference check : " + singleton);  
 System.*out*.println("Object reference check : " + singleton1);  
 }  
  
 public static class Singleton implements Serializable {  
 private static Singleton *ourInstance*;  
  
 public static Singleton getInstance() {  
 if (*ourInstance* == null) {  
 *ourInstance* = new Singleton();  
 }  
 return *ourInstance*;  
 }  
  
 private Singleton() {  
 }  
 }  
}

package com.javarush.task.task20.task2009;  
  
/\*   
Как сериализовать static?  
Сделай так, чтобы сериализация класса ClassWithStatic была возможной.  
\*/  
public class Solution {  
 public static class ClassWithStatic {  
 public static String *staticString* = "it's test static string";  
 public int i;  
 public int j;  
 }  
  
 public static void main(String[] args) {  
  
 }  
}

package com.javarush.task.task20.task2010;  
  
/\*   
Как сериализовать что-то свое?  
Сделайте так, чтобы сериализация класса Object была возможной.  
\*/  
public class Solution {  
 public static class Object {  
 public String string1;  
 public String string2;  
 }  
  
 public static int *countStrings*;  
  
 public static class String {  
 private final int number;  
  
 public String() {  
 number = ++*countStrings*;  
 }  
  
 public void print() {  
 System.*out*.println("string #" + number);  
 }  
 }  
  
 public static void main(String[] args) {  
  
 }  
}

package com.javarush.task.task20.task2011;  
  
/\*   
Externalizable для апартаментов  
Реализуй интерфейс Externalizable в классе Apartment.  
\*/  
public class Solution {  
  
 public static class Apartment {  
  
 private String address;  
 private int year;  
  
 */\*\*  
 \* Mandatory public no-arg constructor.  
 \*/* public Apartment() { super(); }  
  
 public Apartment(String adr, int y) {  
 address = adr;  
 year = y;  
 }  
  
 */\*\*  
 \* Prints out the fields. used for testing!  
 \*/* public String toString() {  
 return("Address: " + address + "\n" + "Year: " + year);  
 }  
 }  
  
 public static void main(String[] args) {  
  
 }  
}

package com.javarush.task.task20.task2013;  
  
import java.io.Externalizable;  
import java.io.IOException;  
import java.io.ObjectInput;  
import java.io.ObjectOutput;  
import java.util.List;  
  
/\*   
Externalizable Person  
Класс Person должен сериализовываться с помощью интерфейса Externalizable.  
Исправь ошибку сериализации.  
Сигнатуры методов менять нельзя.  
\*/  
public class Solution {  
 public static class Person {  
 private String firstName;  
 private String lastName;  
 private int age;  
 private Person mother;  
 private Person father;  
 private List<Person> children;  
  
 public Person(String firstName, String lastName, int age) {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.age = age;  
 }  
  
 public void setMother(Person mother) {  
 this.mother = mother;  
 }  
  
 public void setFather(Person father) {  
 this.father = father;  
 }  
  
 public void setChildren(List<Person> children) {  
 this.children = children;  
 }  
  
 @Override  
 public void writeExternal(ObjectOutput out) throws IOException {  
 out.writeObject(mother);  
 out.writeObject(father);  
 out.writeChars(firstName);  
 out.writeChars(lastName);  
 out.writeInt(age);  
 out.writeObject(children);  
 }  
  
 @Override  
 public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException {  
 firstName = in.readLine();  
 lastName = in.readLine();  
 father = (Person)in.readObject();  
 mother = (Person)in.readObject();  
 age = in.readInt();  
 children = (List)in.readObject();  
 }  
 }  
  
 public static void main(String[] args) {  
  
 }  
}

package com.javarush.task.task20.task2014;  
  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
/\*   
Serializable Solution  
Сериализуй класс Solution.  
Подумай, какие поля не нужно сериализовать, пометь ненужные поля модификатором transient.  
Объект всегда должен содержать актуальные итоговые данные.  
Метод main не участвует в тестировании.  
  
Написать код проверки самостоятельно в методе main:  
1) создать файл, открыть поток на чтение (input stream) и на запись(output stream);  
2) создать экземпляр класса Solution — savedObject;  
3) записать в поток на запись savedObject (убедитесь, что они там действительно есть);  
4) создать другой экземпляр класса Solution с другим параметром;  
5) загрузить из потока на чтение объект — loadedObject;  
6) проверить, что savedObject.string равна loadedObject.string;  
7) обработать исключения.  
\*/  
public class Solution {  
 public static void main(String[] args) {  
 System.*out*.println(new Solution(4));  
 }  
  
 private final String pattern = "dd MMMM yyyy, EEEE";  
 private Date currentDate;  
 private int temperature;  
 String string;  
  
 public Solution(int temperature) {  
 this.currentDate = new Date();  
 this.temperature = temperature;  
  
 string = "Today is %s, and current temperature is %s C";  
 SimpleDateFormat format = new SimpleDateFormat(pattern);  
 this.string = String.*format*(string, format.format(currentDate), temperature);  
 }  
  
 @Override  
 public String toString() {  
 return this.string;  
 }  
}

package com.javarush.task.task20.task2015;  
  
import java.io.IOException;  
import java.io.ObjectInputStream;  
import java.io.ObjectOutputStream;  
import java.io.Serializable;  
  
/\*   
Переопределение сериализации  
Сделайте так, чтобы после десериализации нить runner продолжила работать.  
Ключевые слова объекта runner менять нельзя.  
  
Hint/Подсказка:  
Конструктор не вызывается при десериализации, только инициализируются все поля.  
\*/  
public class Solution implements {  
 private Thread runner;  
 private int speed;  
  
 public Solution(int speed) {  
 this.speed = speed;  
 runner = new Thread(this);  
 runner.start();  
 }  
  
 public void run() {  
 // do something here, does not matter  
 }  
  
 */\*\*  
 Переопределяем сериализацию.  
 Для этого необходимо объявить методы:  
 private void writeObject(ObjectOutputStream out) throws IOException  
 private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException  
 Теперь сериализация/десериализация пойдет по нашему сценарию :)  
 \*/* private void writeObject(ObjectOutputStream out) throws IOException {  
 out.defaultWriteObject();  
 }  
  
 private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException {  
 in.defaultReadObject();  
 }  
  
 public static void main(String[] args) {  
  
 }  
}

package com.javarush.task.task20.task2016;  
  
/\*   
Минимум изменений  
Используя минимум изменений кода сделайте так, чтобы сериализация класса C стала возможной.  
\*/  
public class Solution {  
 public class A {  
 String name = "A";  
  
 public A(String name) {  
 this.name += name;  
 }  
  
 @Override  
 public String toString() {  
 return name;  
 }  
 }  
  
 public class B extends A {  
 String name = "B";  
  
 public B(String name) {  
 super(name);  
 this.name += name;  
 }  
 }  
  
 public class C extends B {  
 String name = "C";  
  
 public C(String name) {  
 super(name);  
 this.name = name;  
 }  
 }  
  
 public static void main(String[] args) {  
  
 }  
}

package com.javarush.task.task20.task2017;  
  
import java.io.ObjectInputStream;  
  
/\*   
Десериализация  
На вход подается поток, в который записан сериализованный объект класса A либо класса B.  
Десериализуй объект в методе getOriginalObject так, чтобы в случае возникновения исключения было выведено сообщение на экран и возвращен null.  
Реализуй интерфейс Serializable где необходимо.  
\*/  
public class Solution {  
 public A getOriginalObject(ObjectInputStream objectStream) {  
 return null;  
 }  
  
 public class A {  
 }  
  
 public class B extends A {  
 public B() {  
 System.*out*.println("inside B");  
 }  
 }  
  
 public static void main(String[] args) {  
  
 }  
}

package com.javarush.task.task20.task2018;  
  
import java.io.Serializable;  
  
/\*   
Найти ошибки  
Почему-то при сериализации/десериализации объекта класса B возникают ошибки.  
Найди проблему и исправь ее.  
Класс A не должен реализовывать интерфейсы Serializable и Externalizable.  
В сигнатуре класса В ошибки нет :).  
Метод main не участвует в тестировании.  
\*/  
public class Solution {  
 public static class A {  
 protected String name = "A";  
  
 public A(String name) {  
 this.name += name;  
 }  
 }  
  
 public class B extends A implements Serializable {  
 public B(String name) {  
 super(name);  
 this.name += name;  
 }  
 }  
  
 public static void main(String[] args) throws IOException, ClassNotFoundException {  
 ByteArrayOutputStream arrayOutputStream = new ByteArrayOutputStream();  
 ObjectOutputStream oos = new ObjectOutputStream(arrayOutputStream);  
  
 Solution solution = new Solution();  
 B b = solution.new B("B2");  
 System.*out*.println(b.name);  
  
 oos.writeObject(b);  
  
 ByteArrayInputStream arrayInputStream = new ByteArrayInputStream(arrayOutputStream.toByteArray());  
 ObjectInputStream ois = new ObjectInputStream(arrayInputStream);  
  
 B b1 = (B)ois.readObject();  
 System.*out*.println(b1.name);  
 }  
}

package com.javarush.task.task20.task2019;  
  
import java.io.\*;  
import java.util.HashMap;  
import java.util.Map;  
  
/\*   
Исправить ошибку. Сериализация  
После десериализации объекта класса Solution обнаружили, что данных в словаре [m] нет :(  
  
Исправить 1 ошибку.  
\*/  
public class Solution implements Serializable {  
  
 public static void main(String args[]) throws Exception {  
 FileOutputStream fileOutput = new FileOutputStream("your.file.name");  
 ObjectOutputStream outputStream = new ObjectOutputStream(fileOutput);  
  
 Solution solution = new Solution();  
 outputStream.writeObject(solution);  
  
 fileOutput.close();  
 outputStream.close();  
  
 //loading  
 FileInputStream fiStream = new FileInputStream("your.file.name");  
 ObjectInputStream objectStream = new ObjectInputStream(fiStream);  
  
 Solution loadedObject = (Solution) objectStream.readObject();  
  
 fiStream.close();  
 objectStream.close();  
  
 //Attention!!  
 System.*out*.println(loadedObject.size());  
 }  
  
 private Map<String, String> m = new HashMap<>();  
  
 public Map<String, String> getMap() {  
 return m;  
 }  
  
 public void Solution() {  
 m.put("Mickey", "Mouse");  
 m.put("Mickey", "Mantle");  
 }  
  
 public int size() {  
 return m.size();  
 }  
}

package com.javarush.task.task20.task2020;  
  
import java.io.PrintStream;  
import java.util.logging.Logger;  
  
/\*   
Сериализуйте Person  
Сериализуй класс Person стандартным способом. При необходимости добавь некоторым полям модификатор transient.  
\*/  
public class Solution {  
  
 public static class Person {  
 String firstName;  
 String lastName;  
 String fullName;  
 final String greetingString;  
 String country;  
 Sex sex;  
 PrintStream outputStream;  
 Logger logger;  
  
 Person(String firstName, String lastName, String country, Sex sex) {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.fullName = String.*format*("%s, %s", lastName, firstName);  
 this.greetingString = "Hello, ";  
 this.country = country;  
 this.sex = sex;  
 this.outputStream = System.*out*;  
 this.logger = Logger.*getLogger*(String.*valueOf*(Person.class));  
 }  
 }  
  
 enum Sex {  
 *MALE*,  
 *FEMALE* }  
  
 public static void main(String[] args) {  
  
 }  
}

package com.javarush.task.task20.task2021;  
  
import java.io.\*;  
  
/\*   
Запрет сериализации  
Запрети сериализацию класса SubSolution используя NotSerializableException.  
Сигнатуры классов менять нельзя.  
\*/  
public class Solution implements Serializable {  
 public static class SubSolution extends Solution {  
 }  
  
 public static void main(String[] args) {  
  
 }  
}

package com.javarush.task.task20.task2022;  
  
import java.io.\*;  
  
/\*   
Переопределение сериализации в потоке  
Сериализация/десериализация Solution не работает.  
Исправь ошибки не меняя сигнатуры методов и класса.  
Метод main не участвует в тестировании.  
  
Написать код проверки самостоятельно в методе main:  
1) создать экземпляр класса Solution  
2) записать в него данные — writeObject  
3) сериализовать класс Solution — writeObject(ObjectOutputStream out)  
4) десериализовать, получаем новый объект  
5) записать в новый объект данные — writeObject  
6) проверить, что в файле есть данные из п.2 и п.5  
\*/  
public class Solution implements Serializable, AutoCloseable {  
 private FileOutputStream stream;  
  
 public Solution(String fileName) throws FileNotFoundException {  
 this.stream = new FileOutputStream(fileName);  
 }  
  
 public void writeObject(String string) throws IOException {  
 stream.write(string.getBytes());  
 stream.write("\n".getBytes());  
 stream.flush();  
 }  
  
 private void writeObject(ObjectOutputStream out) throws IOException {  
 out.defaultWriteObject();  
 out.close();  
 }  
  
 private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException {  
 in.defaultReadObject();  
 in.close();  
 }  
  
 @Override  
 public void close() throws Exception {  
 System.*out*.println("Closing everything!");  
 stream.close();  
 }  
  
 public static void main(String[] args) {  
  
 }  
}

package com.javarush.task.task20.task2023;  
  
/\*   
Правильный вывод  
Расставить обращение к методам суперкласса и модификаторы доступа так, чтобы вывод на экран был следующим:  
  
C class, method2  
A class, method2  
A class, method1  
B class, method1  
  
1. Из одного метода можно вызвать только один метод суперкласса.  
2. Из одного метода можно вызвать только один метод класса.  
3. Можно менять модификаторы доступа к методам.  
\*/  
public class Solution {  
 public static void main(String[] s) {  
 A a = new C();  
 a.method2();  
 }  
  
 public static class A {  
 public void method1() {  
 System.*out*.println("A class, method1");  
 }  
  
 public void method2() {  
 System.*out*.println("A class, method2");  
 }  
 }  
  
 public static class B extends A {  
 public void method1() {  
 System.*out*.println("B class, method1");  
 }  
  
 public void method2() {  
 System.*out*.println("B class, method2");  
 }  
 }  
  
 public static class C extends B {  
 public void method1() {  
 System.*out*.println("C class, method1");  
 }  
  
 public void method2() {  
 System.*out*.println("C class, method2");  
 }  
 }  
}

package com.javarush.task.task20.task2024;  
  
import java.util.LinkedList;  
import java.util.List;  
  
/\*   
Знакомство с графами  
Прочитать в дополнительных материалах о сериализации графов.  
Дан ориентированный плоский граф Solution, содержащий циклы и петли.  
  
Пример, http://edu.nstu.ru/courses/saod/images/graph1.gif  
  
Сериализовать Solution.  
Все данные должны сохранить порядок следования.  
\*/  
public class Solution {  
 int node;  
 List<Solution> edges = new LinkedList<>();  
  
 public static void main(String[] args) {  
  
 }  
}

package com.javarush.task.task20.task2025;  
  
/\*   
Алгоритмы-числа  
Число S состоит из M цифр, например, S=370 и M (количество цифр) = 3  
Реализовать логику метода getNumbers, который должен среди натуральных чисел меньше N (long)  
находить все числа, удовлетворяющие следующему критерию:  
число S равно сумме его цифр, возведенных в M степень  
getNumbers должен возвращать все такие числа в порядке возрастания.  
  
Пример искомого числа:  
370 = 3\*3\*3 + 7\*7\*7 + 0\*0\*0  
8208 = 8\*8\*8\*8 + 2\*2\*2\*2 + 0\*0\*0\*0 + 8\*8\*8\*8  
  
На выполнение дается 10 секунд и 50 МБ памяти.  
\*/  
public class Solution {  
 public static long[] getNumbers(long N) {  
 long[] result = null;  
 return result;  
 }  
  
 public static void main(String[] args) {  
  
 }  
}

package com.javarush.task.task20.task2026;  
  
/\*   
Алгоритмы-прямоугольники  
1. Дан двумерный массив N\*N, который содержит несколько прямоугольников.  
2. Различные прямоугольники не соприкасаются и не накладываются.  
3. Внутри прямоугольник весь заполнен 1.  
4. В массиве:  
4.1) a[i, j] = 1, если элемент (i, j) принадлежит какому-либо прямоугольнику  
4.2) a[i, j] = 0, в противном случае  
5. getRectangleCount должен возвращать количество прямоугольников.  
6. Метод main не участвует в тестировании  
\*/  
public class Solution {  
 public static void main(String[] args) {  
 byte[][] a = new byte[][]{  
 {1, 1, 0, 0},  
 {1, 1, 0, 0},  
 {1, 1, 0, 0},  
 {1, 1, 0, 1}  
 };  
 int count = *getRectangleCount*(a);  
 System.*out*.println("count = " + count + ". Должно быть 2");  
 }  
  
 public static int getRectangleCount(byte[][] a) {  
 return 0;  
 }  
}

package com.javarush.task.task20.task2027;  
  
import java.util.List;  
  
/\*   
Кроссворд  
1. Дан двумерный массив, который содержит буквы английского алфавита в нижнем регистре.  
2. Метод detectAllWords должен найти все слова из words в массиве crossword.  
3. Элемент(startX, startY) должен соответствовать первой букве слова, элемент(endX, endY) — последней.  
text — это само слово, располагается между начальным и конечным элементами  
4. Все слова есть в массиве.  
5. Слова могут быть расположены горизонтально, вертикально и по диагонали как в нормальном, так и в обратном порядке.  
6. Метод main не участвует в тестировании.  
\*/  
public class Solution {  
 public static void main(String[] args) {  
 int[][] crossword = new int[][]{  
 {'f', 'd', 'e', 'r', 'l', 'k'},  
 {'u', 's', 'a', 'm', 'e', 'o'},  
 {'l', 'n', 'g', 'r', 'o', 'v'},  
 {'m', 'l', 'p', 'r', 'r', 'h'},  
 {'p', 'o', 'e', 'e', 'j', 'j'}  
 };  
 *detectAllWords*(crossword, "home", "same");  
 /\*  
Ожидаемый результат  
home - (5, 3) - (2, 0)  
same - (1, 1) - (4, 1)  
 \*/  
 }  
  
 public static List<Word> detectAllWords(int[][] crossword, String... words) {  
  
 return null;  
 }  
  
 public static class Word {  
 private String text;  
 private int startX;  
 private int startY;  
 private int endX;  
 private int endY;  
  
 public Word(String text) {  
 this.text = text;  
 }  
  
 public void setStartPoint(int i, int j) {  
 startX = i;  
 startY = j;  
 }  
  
 public void setEndPoint(int i, int j) {  
 endX = i;  
 endY = j;  
 }  
  
 @Override  
 public String toString() {  
 return String.*format*("%s - (%d, %d) - (%d, %d)", text, startX, startY, endX, endY);  
 }  
 }  
}