# 基本概念

序列化和反序列化是java总进行数据存储和数据传输的一种方式。

1. 对象的序列化：对象转化为字节
2. 对象的反序列化：字节转化为对象

说明：有时也将对象转化为字符串的过程也称为序列化，例如将对象转化为json字符串。

# Java中如何实现序列化和反序列化

1. 对象之间或间接实现Serializable接口
2. 添加序列化id(为反序列化提供保障)
3. 借助对象流对象实现小老虎和反序列化

class Message implements Serializable {  
 */\*此序列化ID保证了以后如果此类发生结构变化（例如加个属性）  
 \* 已经序列化的内容被反序列化时不会出错\*/* private static final long *serialVersionUID* = 2668788440691836642L;  
 private int id = 0;  
 private String context = "TestSerializable";  
 private String name = "Clark";  
  
 @Override  
 public String toString() {  
 return "Message{" +  
 "id=" + id +  
 ", context='" + context + '\'' +  
 ", name='" + name + '\'' +  
 '}';  
 }  
}  
  
public class Test01 {  
 static void serialiable(Object o,File file) throws IOException{  
 var fos = new FileOutputStream(file);  
 ObjectOutputStream oos = new ObjectOutputStream(fos);  
 oos.writeObject(o);  
 System.*out*.println("Serializable sucessful");  
 oos.close();  
 }  
 static Object deseSerializable(Object o,File file) throws IOException, ClassNotFoundException {  
 FileInputStream fis = new FileInputStream(file);  
 ObjectInputStream ois = new ObjectInputStream(fis);  
 Object o2 = ois.readObject();  
 System.*out*.println("Deseserialiable sucessful");  
 ois.close();  
 return o2;  
 }  
 public static void main(String[] args) throws Exception {  
 File file = new File("D:\\Java\\a.txt");  
 Message msg = new Message();  
 *//序列化  
// serialiable(msg,file);  
 //反序列化* Object obj = *deseSerializable*(msg,file);  
 *//输出反序列化后的对象* System.*out*.println(obj);  
 }  
}

# 设计一个SerializableCache（参考Mybatis）

public class SerializableCache implements Cache{  
 private Cache cache;  
 public SerializableCache(Cache cache){  
 this.cache = cache;  
 }  
 *//将对象序列化存到字节数组中* private byte[] serializable(Object o) throws IOException {  
 *//1.构建字节数组输出流对象* ByteArrayOutputStream baos = new ByteArrayOutputStream();  
 *//2.构建对象输出流* ObjectOutputStream oos = new ObjectOutputStream(baos);  
 *//3.执行对象序列化* oos.writeObject(o);  
 *//4.释放资源* oos.close();  
 baos.close();  
 *//5.返回字节数组* return baos.toByteArray();  
 }  
 *//将字节数组中的字节反序列化为对象* private Object deserializable(byte[] arr)throws Exception{  
 *//1.构建字节数组输入流对象* ByteArrayInputStream bais = new ByteArrayInputStream(arr);  
 *//2.构建对象输入流对象* ObjectInputStream ois = new ObjectInputStream(bais);  
 *//3.将字节数组内容读取为对象* Object obj = ois.readObject();  
 *//4.释放资源* ois.close();  
 bais.close();  
 *//5.返回对象* return obj;  
 }  
 @Override  
 public int size() {  
 return cache.size();  
 }  
 @Override  
 public void putObject(Object key, Object val) {  
 try {  
 *//1.将对象序列化(字节数组)* byte[] arr = serializable(val);  
 *//2.将对象存储到cache* cache.putObject(key, arr);  
 }catch (IOException e){  
 e.printStackTrace();  
 *//抛出异常* throw new RuntimeException(e);  
 }  
 }  
 @Override  
 public Object getObject(Object key) {  
 *//1.从cache获取对象(字节数组)* byte[] val = (byte[])cache.getObject(key);  
 try {  
 *//2.将字节数组中的内容反序列化* Object obj = deserializable(val);  
 return obj;  
 }catch (Exception e){  
 e.printStackTrace();  
 *//抛出异常* throw new RuntimeException(e);  
 }  
 }  
 @Override  
 public Object removeObject(Object key) {  
 return cache.removeObject(key);  
 }  
 @Override  
 public void clear() {  
 cache.clear();  
 }  
  
 @Override  
 public String toString() {  
 return cache.toString();  
 }  
}