1. What Kind of Object is System.out?

Using BlueJ or a Java class, find the class type of System.out and print it.
Use System.out.getClass().getName().

2. Copy a File byte-by-byte

This problem shows basic use of InputStream, but its inefficient.
2.1 Complete this code

```
public class FileUtil {
    /**
     * Copy source file to destination.
     * @param source is the source. Must be a readable, plain file.
     * @param target is the destination of the copy.
     * @return reference to the output file.
     */
    public static File copyfile(File source, File target)
            throws IOException {
        //TODO throw IllegalArgumentException if any of these:
        // 1) source does not exists
        // 2) source is not a plain file (e.g. directory, special file)
        // 3) source is not reable
        if ( ! file.exists() ) throw _____________________________

        // copy file byte by byte
        InputStream in = new FileInputStream(source);
        OutputStream out = new FileOutputStream(target);
        do {
            int b = in.read();
            if (b < 0) break;  // no more input
            out.write(b);
        } while( b>= 0);
        //TODO close the in and out streams
        return target;
    }
}
```

2.2 Find a JPG or PNG file of size 5-10MB. Copy it and print the amount of time used.

After you copy the file, look at the copied image to verify it was copied correctly.

```
public static void main(String [] args) {
    File src = new File( "/path/to/image.jpg" );
    File dest = new File( /*TODO*/ );
    long start = System.nanoTime( );
    File result = copyFile(src, dest);
    long stop = System.nanoTime( );
    System.out.printf("Copied %,d bytes in %.6f seconds\n",
            result.length(), elapsed(start,stop) );
}

public static double elapsed(long start, long stop) {
    //TODO compute elapsed time and convert to seconds
}
```

3. Copy a FIle using block reads (read to array of byte)

Its much faster to read and write data in blocks instead of byte-by-byte.

3.1 Write another `copyfile` method: `copyfile(source, target, buffsize)`

that uses these methods:

InputStream

`int read( byte[ ] b )` - read inputstream data into an array of bytes. The return value is the number of bytes actually read, which may be smaller than the array size! Returns -1 at end of input.

OutputStream

`void write( byte[ ] b, int offset, int length )`- write bytes from array (b) to output stream, starting at index `offset`, and wrting `length` bytes.

```
public static File copyfile(File source, File target, int buffsize)
          throws IOException {
    // statements similar to previous method
    byte[] b = new byte[buffsize];
    do {
        int size = in.read(b);
        if (size <= 0) break;  // no more input
        out.write( b, _______, _______);
    } while(size>0);
```

3.2 Copy the same image as in the previous problem. Try buffer sizes 4*1024, 16*1024, 1024*1024. Which is fastest?

Note: if you modify this code to copy a file from a URL or socket (network copy), then the optimal buffer size will be smaller than for local files.