*CSE 102*

**Files-II**

1. What is the output of the following program? Explain yourself why.

**public** **static** **void** main(String[] args) **throws** Exception{

ObjectOutputStream os = **new** ObjectOutputStream(**new** FileOutputStream("file.dat"));

os.write(0xFFFFFFFF);

os.close();

ObjectInputStream is = **new** ObjectInputStream(**new** FileInputStream("file.dat"));

System.***out***.println(is.read() + ", " + is.read());

}

1. 255, -1
2. 255, 255
3. 256, 256
4. 0, 0
5. What is the output of the following program? Explain yourself why.

**public** **static** **void** main(String[] args) **throws** Exception{

ObjectOutputStream os = **new** ObjectOutputStream(**new** FileOutputStream("file.dat"));

os.writeInt(0xFFFFFFFF);

os.close();

ObjectInputStream is = **new** ObjectInputStream(**new** FileInputStream("file.dat"));

System.***out***.println(is.read() + ", " + is.read());

}

1. 255, -1
2. 255, 255
3. 256, 256
4. 0, 0
5. What is the output of the following program?

**public** **static** **void** main(String[] args) **throws** Exception{

ObjectOutputStream os = **new** ObjectOutputStream(**new** FileOutputStream("file.dat"));

os.writeBoolean(**false**);

os.close();

ObjectInputStream is = **new** ObjectInputStream(**new** FileInputStream("file.dat"));

System.***out***.println(is.read());

}

1. Runtime error, because read() method reads a byte but we wrote a boolean to the file.
2. 0, because under the hood writeBoolean() method writes a single byte of value 0 to represent a false value.
3. false
4. -1, because a writeBoolean() writes a single bit (not a byte) to the file but read() tries to read a full byte.
5. What is the output of the following program? Why?

**public** **static** **void** main(String[] args) **throws** Exception{

ObjectOutputStream os = **new** ObjectOutputStream(**new** FileOutputStream("file.dat"));

os.write(0);

os.write(0);

os.write(1);

os.write(0);

os.close();

ObjectInputStream is = **new** ObjectInputStream(**new** FileInputStream("file.dat"));

System.***out***.println(is.readInt());

}

1. 0010
2. 10
3. 0
4. 256
5. Run the following program and observe the result. What would happen if we add one more integer field to the Bundle class?

**class** Bundle **implements** Serializable {

**int** a, b, c;

**boolean** x, y;

**float** m, n;

}

**public** **class** Main {

**public** **static** **void** main(String[] args) **throws** Exception{

ObjectOutputStream os = **new** ObjectOutputStream(**new** FileOutputStream("file.dat"));

File f = **new** File("file.dat");

**long** size = f.length();

Bundle b = **new** Bundle();

**for**(**int** i=0; i<10; i++) {

os.writeObject(b);

System.***out***.println(f.length() - size);

size = f.length();

}

os.close();

}

}

1. We end up writing more bytes to the file in each iteration of the loop since the size of a single object is increased.
2. Nothing will change.
3. Only the first call to the writeObject() method will cause more bytes to be written.
4. Only the last call to the writeObject() method will cause more bytes to be written.
5. Write a program which takes the absolute path of a directory and computes the total size of the directory (including all its subdirectories) using recursion. Methods of File class will be helpful.
6. Write the same program but instead of recursion, use a queue.
7. Write a program which reports all files in a given directory whose size is greater than a threshold value.

**ANSWERS:**

1. A
2. B
3. B
4. D
5. C

**public** **static** **void** main(String[] args) {

File f = **new** File("dirPathHere");

System.***out***.println(*getSize*(f));

}

**static** **long** getSize(File f) {

**if**(!f.isDirectory())

**return** f.length();

**else** {

**long** childrenSum = 0;

**for**(File ch: f.listFiles())

childrenSum += *getSize*(ch);

**return** childrenSum;

}

}

**static** **long** getSize(File f) {

**long** res = 0;

Queue<File> q = **new** LinkedList<>();

q.add(f);

**while**(!q.isEmpty()) {

File cur = q.remove();

**if**(cur.isDirectory())

**for**(File ch: cur.listFiles())

q.add(ch);

**else**

res += cur.length();

}

**return** res;

}

**static** **void** reportBigs(File f, **long** threshold) {

**int** pathLen = f.getAbsolutePath().length();

Queue<File> q = **new** LinkedList<>();

q.add(f);

**while**(!q.isEmpty()) {

File cur = q.remove();

**if**(cur.isDirectory())

**for**(File ch: cur.listFiles())

q.add(ch);

**else** **if**(cur.length() > threshold) System.***out***.println(cur.getAbsolutePath().substring(pathLen));

}

}