

Reduce JAR Size

This article explains options for reducing [JAR](#) size if your application becomes too big to install on the device. Note that this article was submitted by atiskov as an entry to the St Petersburg Developer Day competition.

Overview

Sometimes a [JAR](#) file that contains your application becomes too big to install on the device.

There are two ways of reducing final JAR file:

- Reduce the size of resources that are included in JAR file (pictures, binary files, text resources etc.)
- Reduce the size of class files (files that contain code of your application)

Taking the first approach, you should:

- Use special algorithms that compress text data in your text-files (maybe Huffman algorithm). But at the same time you should place special code in your java-files in order to decode such text-files, and your JAR size can become larger!
- Use pictures that do not contain more colors than mobile-phone supports (often there is no need to store pictures with 24bit depth colors) Try reducing image colour depth (for example using GIMP or Photoshop). See for more information.
- Use a picture resolution that will be small enough to fit your application requirements (no need to store pictures 1280*1024...)

Taking the second approach - there are two types of actions in order to reduce size of class-files:

- Actions that always will help you to reduce size of class-files (less or more)
- Actions that sometimes reduce size of class-files (but they also can increase it!) depending on the situation (code of classes in your application)

First Type - "Less is More"

- Using obfuscators. Note that on different projects different obfuscators can show very unlike results.
 - Use different settings of obfuscating process that your current obfuscator allows.
 - Sometimes it is possible to indicate which ZIP - archive program to use (JAR file is a ZIP archive).
 - Try different settings of this archive program
- Make variables, classes, methods and class members final. Constants require less space for storing in class-files (use modifier "final").
- Erase all your System.out.print's if they don't be valuable for user in release version of your application or introduce variable

```
public static final boolean LOG_ENABLED = true;
```

And write

```
if (LOG_ENABLED)
/*...*/
```

above invocations of System.out.print's. And before you will make a release version of your application, set LOG_ENABLED to false and use obfuscator (he will cut off stuff from class files)

- Delete unnecessary parameters of methods (that aren't used during method's flow)

Second type

- Try different visibility modifiers of members and methods (static, public, private).
- Try not to use inner classes.
- You can inline some methods (especially those which are invoked only once per application run), variables, class members and classes ([Reduce JAR Size: Replacing classes with arrays](#)).
- Use variables of less size (e.g. 'byte' occupies 8 bit and "long" occupies 64 bit).
- If it is possible try to combine all variables or class members of the same type in the array ([Reduce JAR Size: Replacing classes with arrays](#)).

- Initialize your arrays by values that are stored not in code, but in resource files (e.g. text data).
- Use low-level API when drawing simple things (e.g. do not store white background in 16bit-colored picture in PNG-format – simply use `Graphics.fillRect` method with parameters of width and height of screen and appropriate color).
- Sometimes it is possible to replace method by a constant – do this. (e.g. value of screen resolution can be stored in the final class members).
- Replace "switch" conditions by "if-else" or even simply "if".