

**CS 1110, LAB 13: SUBCLASSES; EXCEPTIONS**  
<http://www.cs.cornell.edu/courses/cs1110/2014sp/labs/lab13.pdf>

L. LEE AND S. MARSCHNER

**First Name:** \_\_\_\_\_ **Last Name:** \_\_\_\_\_ **NetID:** \_\_\_\_\_

**Files to download.** Create a *new* directory on your hard drive and download the following modules into that directory.

card.py (<http://www.cs.cornell.edu/courses/cs1110/2014sp/labs/lab13/card.py>)

extendedcard.py (<http://www.cs.cornell.edu/courses/cs1110/2014sp/labs/lab13/extendedcard.py>)

**Getting credit for lab completion.** When done, show your code and/or this handout to a staff member, who will ask you a few questions to see that you understood the material and then swipe your ID card to record your success. Please check the followings.

- You are having a copy of this handout with all of the white boxes filled up at your best.
- You have your completed extendedcard.py open in Komodo Edit.
- You have a command shell open in the directory your code is in, so you can demonstrate the running of your code.

As always, if you do not finish during the lab, **you have until the end of consulting hours on Monday May 5 to finish it.** But you should always do your best to finish during lab hours. Remember that labs are graded on effort, not correctness.

---

EXTENDING CLASS CARD TO HANDLE JOKERS

You may recall our class `Card` from previous labs, which represents “standard” cards from the usual 52-card deck; see the file `card.py` in this lab’s files. In today’s lab, we’ll use subclassing to let us include red and black jokers without having to repeat the code we wrote for `Card`. In particular, we’ll add a new suit, *Joker*, but not add new ranks; instead, we’ll have the black joker have rank 1 and the red joker have rank 2.

Take a look at the skeleton file `extendedcard.py`. In lines 16–19, you can see how we’ve begun to set up the new “Joker” suit using class variables in the subclass `ExtendedCard`.

0.1. **Understanding the skeleton file.** Let's do a few finger exercises to understand various aspects of the skeleton file.

Open up a command-line interface in the same directory as you have the lab files. Start up Python, and then at the Python interactive prompt import the module `extendedcard`.

Now, try the following:

```
extendedcard.ExtendedCard.SUIT_NAMES
```

You should get a list of five suits, including "Joker" at the end.

Now try the following:

```
extendedcard.SUIT_NAMES
```

You should get the error "AttributeError: 'module' object has no attribute 'SUIT\_NAMES'." Why?

Make sure you understand the difference between `extendedcard` and `ExtendedCard` in this lab; if you need help with this, ask now.

Now, quit Python, and then change line 12 of `extendedcard.py` so that instead of "card.Card" as the parent class of `ExtendedCard`, you put "Card". Restart Python, and re-import module `extendedcard`. Uh oh; you (should) get an error; why?

Quit Python, and change line 12 back to what it should be.

Next exercise: observe that nowhere in `extendedcard.py` is there an assignment to a variable `RANK_NAMES`. Given this, predict what will happen when you restart Python and then type:

```
import extendedcard
extendedcard.ExtendedCard.RANK_NAMES
```

Now try it. Why *don't* you get an error; where did the value for `RANK_NAMES` come from?

0.2. **Write the `__init__` method for `ExtendedCard`.** For now, let's have the initialization of our new, Joker-enabled cards rely completely on the method we already wrote for initializing regular old cards, which, after all, sets the instance's suit and rank attributes appropriately, assuming it receives appropriate input.

Implement `__init__` for `ExtendedCard` with a single line that calls the `__init__` method of class `Card` in an appropriate fashion.

To test, restart Python, and type in the following:

```
import extendedcard
c = extendedcard.ExtendedCard(1,4)
joker = extendedcard.ExtendedCard(4,1)
```

Now try printing those cards. Note that since you have no `__str__` method in class `ExtendedCard`, what is called is the inherited `__str__` method in class `Card`; the string that method returns is what is printed.

```
print c
print joker
```

You should *not* get an error for the first statement. Why do you get an error for the second statement?

0.3. **Write the `__str__` method for `ExtendedCard`.** We've just seen that we need to customize `__str__` for `ExtendedCard` in order to handle the jokers. Find the commented-out header and docstring for `__str__` in `ExtendedCard`'s definition (it should be around line 69). Uncomment them, and then implement the function.

For the case when the extended card is not a joker, you should call the `__str__` method of the superclass, since that already does the right thing.

You should use the class variables `BJRANK` and `RJRANK` instead of the "magic numbers" 1 and 2 to check the ranks in the case of jokers. See the function `full_deck` near the bottom of file `extendedcard.py` for an example of how `BJRANK` and `RJRANK` can be referenced.

Test your code by running the same test described in the previous subsection (after restarting Python). This time, you should get the printout 'Black Joker' for the variable `joker`.