marker_trait_attr

The tracking issue for this feature is: #29864

Normally, Rust keeps you from adding trait implementations that could overlap with each other, as it would be ambiguous which to use. This feature, however, carves out an exception to that rule: a trait can opt-in to having overlapping implementations, at the cost that those implementations are not allowed to override anything (and thus the trait itself cannot have any associated items, as they're pointless when they'd need to do the same thing for every type anyway).

```
#![feature(marker_trait_attr)]

#[marker] trait CheapToClone: Clone {}

impl<T: Copy> CheapToClone for T {}

// These could potentially overlap with the blanket implementation above,
// so are only allowed because CheapToClone is a marker trait.
impl<T: CheapToClone, U: CheapToClone> CheapToClone for (T, U) {}
impl<T: CheapToClone> CheapToClone for std::ops::Range<T> {}

fn cheap_clone<T: CheapToClone>(t: T) -> T {
    t.clone()
}
```

This is expected to replace the unstable `overlapping_marker_traits` feature, which applied to all empty traits (without needing an opt-in).